

# SHARP®



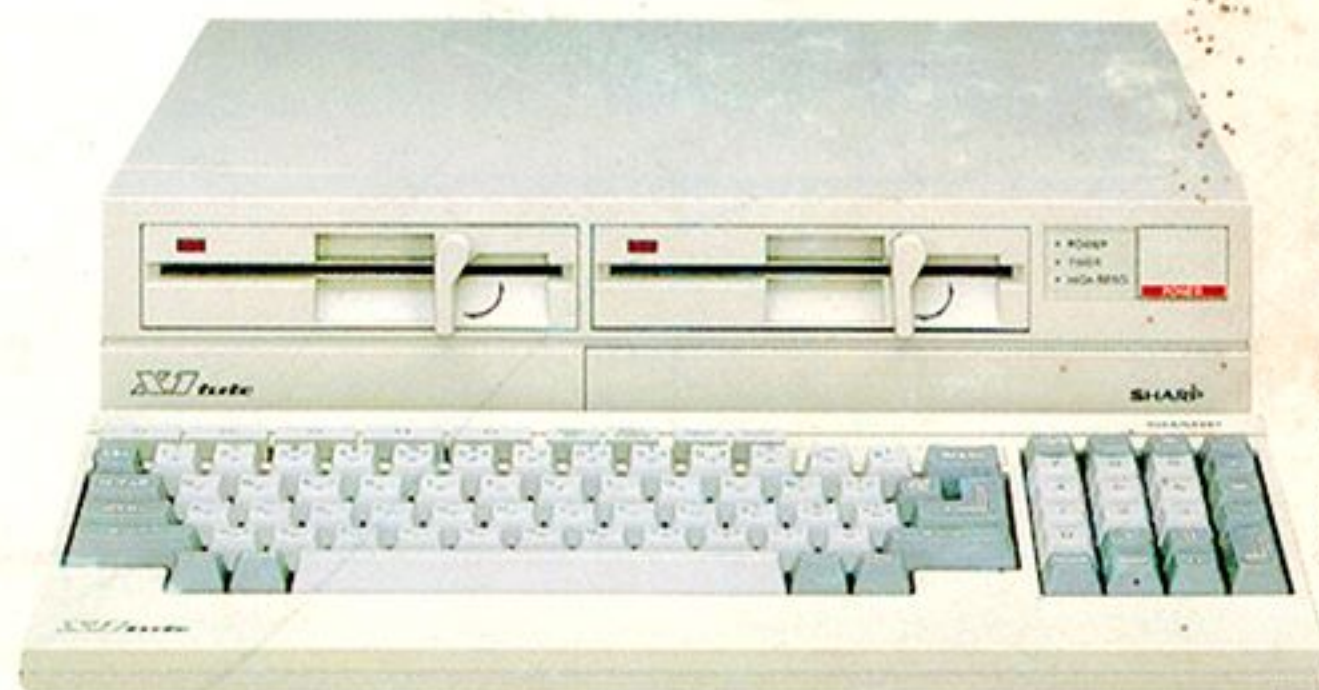
## USER'S MANUAL

パソコンテレビ **AV turbo**  
パーソナルコンピュータ

形 名

# CZ-856C

上手に使って上手に節電



製造番号は、品質管理上重要なものですから商品本体に表示されている製造番号と、保証書に記載されている製造番号とが一致しているか、お確かめください。















# ■USER'S MANUAL(ユーザーズマニュアル)

## はじめに

このマニュアルはBASICの初歩から、本機の特長であるグラフィック機能、日本語処理機能、デジタルテロップ機能などを使いこなしていただくため項目別に説明しています。

目的に応じて各章をお読みください。

入門編…パソコンは初めてという方のため、簡単なプログラムを中心に説明しています。

応用編…入門編が理解でき、さらに使いこなしていただくために説明しています。

なお、本機を正しくご使用いただくため、まず取扱説明書からお読みください。

## ◎ディスクBASIC(CZ-8FB02)を起動させた場合

ディスクBASICが起動されると自動的にStart upというプログラムが実行され、各種の初期設定が行なわれます。

初期設定には、ディスクBASICが起動されるときにすでに行なわれているものと、Start upプログラム中に行なわれているものがあります。

ディスクBASIC起動時の初期設定は変更できませんが、Start upプログラム中に行なわれるものは、このプログラムを自分の希望するように書き換えておけば、初期状態を自由に決めることができます。

### (i) ディスクBASIC起動時の初期状態

INIT

WIDTH 80, 25, CONSOLE 0, 24, 0, 80

(高解像度ディスプレイモードのとき)

WIDTH 80, 12, CONSOLE 0, 11, 0, 80

(標準ディスプレイモードのとき)

KMODE 1

OPTION SCREEN 1

CLEAR &HF400

CLICK ON

REPEAT ON

### (ii) Start upで設定するもの

CLEAR &HF000 (音訓辞書使用時)

KLIST 1

FUNCTION KEYの設定

内蔵時計の「年」の項の設定

### (iii) NEWONの設定

NEWON命令はBASICのコマンドやステートメントなどを削り、フリーエリアを増やします。

ディスクBASICを起動した直後、


NEWON ■


↑

カーソル




と画面に表示されます。

0～9までの数値を入力するか、または数値を省略してリターンキー（）を押してください。希望のレベルのNEWONが設定できます。

NEWON 

とキー入力すれば、すべてのコマンド、ステートメントが使用できます。

NEWONのレベル（0～9）については、応用編「6章  NEWONとフリーエリア」をお読みください。

◎従来のX1用BASIC(CZ-8CB01、CZ-8FB01)とX1turbo用BASIC(CZ-8FB02)との初期設定の相違点

CZ-8CB01、CZ-8FB01	CZ-8FB02
WIDTH 40,25 WIDTH 40,25 CONSOLE 0,25,0,40 CONSOLE 0,25,0,40 KLIST 0の状態 KMODE 0の状態 CLEAR &HF 00	WIDTH 80,25 (高解像度ディスプレイモードの時) WIDTH 80,12 (標準ディスプレイモードの時) CONSOLE 0,24,0,80 (高解像度ディスプレイモードの時) CONSOLE 0,11,0,80 (標準ディスプレイモードの時) KLIST 1 KMODE 1 CLEAR &HF 400

X1turbo用BASICは、以上の点で、従来のX1用BASICと異なっていますので、X1turbo用BASIC上で従来のX1用のソフトウェアを実行する際には、次の操作を行なってください。

CONSOLE 0,25:KLIST0:KMODE0:CLEAR &HF 400 

ただし、X1用のソフトで&HF 400以降を使用しているソフトは利用できません。

なお、本機にはX1用BASIC(CZ-8FB01 V1.0、CZ-8CB01 V1.0)が搭載されています。この起動法については『アプリケーションソフトの説明書』の「はじめに」の項をお読みください。



## 第1部 入門編

### 1章 プログラミングを始めよう

1. コンピュータへのメッセージ	1
2. まずは簡単なプログラム作りから	2
3. プログラムの仕組み	4
3.1 直接実行と間接実行のちがい	4
3.2 プログラムって何だろう	4
3.3 変数を使う	5
3.4 困ったときに助けてくれる <b>CTRL</b> + <b>○</b> キー	5
4. プログラムの作成(その1)	6
4.1 プログラムの入力	6
4.2 1行の挿入	7
4.3 1行の削除	8
4.4 1行の変更	8
4.5 行番号のつけなおし	9
4.6 複数行の削除	10
5. プログラムの作成(その2)	11
5.1 カーソルの移動と文字の修正	11
5.2 効率のよい入力方法	12
5.3 文字の挿入	14
5.4 文字の削除	15

### 2章 絵を描く、色を塗る

1. 点をうつ	17
2. 直線を描く	19
3. 円を描く	20
4. 多角形を描く	21
5. 色を塗る	21

### 3章 プログラムの保存と再生

1. フロッピーディスクに保存、再生する場合	23
2. カセットテープに保存、再生する場合	24
3. フロッピーディスクまたはカセットテープの内容を見るには	26

### 4章 テレビコントロール

1. テレビタイマーコントロール (TV Timer Control)	27
1.1 TV Timer Controlの呼びだし	27
1.2 TV Timer Controlの表示内容	28
1.3 クロック(時計)を現在時刻に合わせる	29
1.4 タイマーで番組予約をする	29
1.5 タイマーでスイッチをOFFにする	30
1.6 タイマーを取消す	30
2. プログラムでテレビをコントロールする	31

### 5章 漢字を表示する

1. 日本語入力	34
2. ミニ・ワープロ機能	44
3. 文字のコピー機能: <b>COPY</b> キー	45

## 第2部 応用編

### 1章 プログラムの編集について

1. 基本的な編集機能	49
2. 知っているると便利な編集機能	50
3. 全角文字の編集	53
4. コントロールコード表	56

### 2章 ディスプレイモード

1. テキスト画面	58
2. グラフィック画面	59
3. カラーコード	64
4. パレットコード	65
5. 中間色コード	65
6. カラーモード	67
7. マルチページモード	67
8. ディスプレイ画面上の座標系	68
8.1 テキスト座標系	68
8.2 グラフィック座標系	70
8.3 ユーザー座標系と画面座標系	72

### 3章 テキスト

1. テキスト画面	75
1.1 ファンクションキーの内容表示	76
1.2 漢字とセミグラフィックパターン	76
2. テキストの属性	77
2.1 色の指定	77
2.2 反転文字	77
2.3 点滅文字	78
2.4 倍文字	78
2.5 ROMCGとRAMCG	80
2.6 アンダーライン	80
3. グラフィックと文字表示	81

### 4章 グラフィックス

1. 画面の初期化	82
2. グラフィックスステートメントの座標指定	83
3. グラフィックスステートメントの色指定	83
4. 線を描く	85
4.1 線を引く	85
4.2 長方形を描く	87
4.3 長方形を塗りつぶす	87
4.4 折れ線を描く	88
5. 正多角形を描く	88
6. 円を描く	91
7. 色を塗る(中間色の指定)	92
8. 色を瞬時に変える	93
9. テキスト文字とグラフィック画面の優先順位を決める	94
10. 文字パターンの描画	95
11. GET@とPUT@	96
12. グラフィックパターンの描画	97



## 5章 日本語処理

- 1. 全角文字とは..... 98
- 2. 全角文字の入力方式..... 99
- 3. 入力モードの選択..... 100
  - 3.1 **HELP** キーについて..... 101
  - 3.2 各入力モードの選択..... 101
- 4. 漢字変換方式について..... 103
  - 4.1 コード入力方式..... 103
  - 4.2 一字変換方式..... 105
  - 4.3 音訓変換方式..... 107

## 6章 フリーエリアについて

- 1. フリーエリア..... 111
  - 1.1 BASICの起動直後..... 111
  - 1.2 プログラムロード後(実行前)..... 111
  - 1.3 プログラム実行後..... 112
  - 1.4 グラフィックVRAMを変数として  
使用しない場合..... 112
- 2. オプションスクリーンとフリーエリア..... 114
- 3. グラフィック描画と外部記憶..... 115
- 4. 日本語入力とフリーエリア..... 116
- 5. NEWONとフリーエリア..... 116
- 6. フリーエリアの大きさを確認するには..... 118

## 7章 ファイルについて

- 1. ファイルの種類..... 119
  - 1.1 シーケンシャルアクセスファイル..... 119
  - 1.2 ランダムアクセスファイル..... 119
- 2. デバイス..... 119
- 3. ファイルの管理..... 120
  - 3.1 ディレクトリ..... 120
  - 3.2 単層ディレクトリ..... 121
  - 3.3 階層ディレクトリ..... 121
  - 3.4 ディレクトリの作成方法..... 122
  - 3.5 カレントディレクトリの変更..... 124
  - 3.6 ディレクトリの削除..... 126
- 4. ファイルの指定..... 126
- 5. デバイス名..... 127
- 6. パス名..... 128
- 7. ファイル名..... 128
- 8. ファイル番号..... 129

## 8章 プログラム、データファイルの保存と再生

- 1. プログラムファイル..... 130
  - 1.1 プログラムの保存..... 130
  - 1.2 プログラムの再生..... 132
  - 1.3 プログラムのペリファイ..... 134
  - 1.4 ファイルの削除..... 134
  - 1.5 ファイル名の一覧表..... 135
  - 1.6 デバイス名の省略..... 136
  - 1.7 プログラムのマージ..... 136
  - 1.8 プログラムのチェイン..... 137
  - 1.9 ファイル名の付けかえ..... 139
  - 1.10 ファイルの属性指定..... 139
  - 1.11 ファイルのパスワード..... 140

- 2. データファイル..... 141
  - 2.1 シーケンシャルファイルへの  
データの保存・再生..... 142
  - 2.2 シーケンシャルファイルへの  
データの追加..... 144
  - 2.3 ランダムアクセスファイルへの  
データの保存・再生..... 145
  - 2.4 ランダムアクセスファイルに関する  
データの追加・変更..... 148

## 9章 ミュージック

- 1. ブザー音を出す..... 151
- 2. 楽譜を演奏する..... 151
- 3. PSGのコントロール..... 153

## 10章 配列

- 1. 配列名と添字..... 158
- 2. 配列宣言..... 159
- 3. **OPTION BASE**..... 159
- 4. 配列変数の消去..... 160
- 5. **VDIM**..... 160

## 11章 機械語サブルーチンとモニタ

- 1. 機械語サブルーチンの入力..... 162
  - 1.1 機械語サブルーチンの記憶領域の設定..... 162
  - 1.2 機械語サブルーチンの入力方法..... 163
- 2. **BASIC**プログラムから機械語  
サブルーチンを呼び出す方法..... 164
  - 2.1 **CALL**ステートメント..... 164
  - 2.2 **USR**関数..... 164
- 3. 機械語モニタ..... 167

## 12章 ディスクの使い方

- 1. ディスクの使用準備..... 177
- 2. ドライブ接続構成..... 177
- 3. システムプログラム起動方法..... 178
- 4. 接続ドライブのディスクタイプ設定..... 181
- 5. ディスクのフォーマット構成..... 183
- 6. ディスクファイル管理方法..... 187

## 13章 RS-232Cインターフェイスの使い方

- 1. 接続方法..... 191
- 2. RS-232Cインターフェイスの  
データ信号フォーマット..... 194
- 3. **BASIC**でRS-232Cインター  
フェイスを取り扱う方法..... 194
- 4. 入出力命令と割り込み処理..... 197
  - 4.1 RS-232Cインターフェイスに  
関する入出力命令..... 197
  - 4.2 割り込み処理..... 198
- 5. ユーザーがマシン語でRS-232C用  
ソフトを作成する場合の使用法..... 200
- 6. 割り込み処理の使用..... 203



<b>14章</b>	<b>マウスインターフェイスの使い方</b>	
1.	マウスとは	204
2.	マウスを使う	204
2.1	マウス関数の書式と機能について	205
2.2	マウス関数を動かす	207
2.3	マウスカーソルを表示させる	208
3.	使用上の注意事項	210

<b>15章</b>	<b>デジタルテロップの使い方</b>	
1.	特長	211
2.	各部名称	211
3.	VTR録画モードスイッチ	212
4.	使用方法	213
5.	黒抜き表示	215
6.	注意事項	215

## 付録

A 1.	テキスト画面(V-RAM)	
	へのアクセス	223
A 2.	組み込み関数以外の数学的関数	225
A 3.	数値精度の変換	226
A 4.	数値データの誤差	227
A 5.	メモリマップ	230
A 6.	ユーザー定義文字の作りかた	232





# 第 1 部

## 入門編

# プログラミングを始めよう

## 1

## コンピュータへのメッセージ

本機に電源を入れBASICをロードします。  
(取扱説明書第2章の「動かしてみる」を参照)  
ディスクBASICを起動すると、NEWON■と表示します。(図1)

NEWON■ と表示したら↵(キャリッジリターンキー)を押してください。

すると、図2のようになります。

画面に四角くて明滅するものがあります。  
このチカチカするものをカーソルといいます。  
カーソルは表示位置を示し、コンピュータがあなたからのメッセージを待っています。

では、手はじめに、キーを

**C** **L** **S**

の順に押してください。画面は図3のようになります。

画面には大文字で表示されていますが、小文字を使ってもかまいません。

(大文字を使うときは **caps lock** キーを押して)  
(ロック状態にします。)

次に、↵(キャリッジリターンキー)キーを押してみてください。画面に書かれていた文字がすべて消え、図4のように表示されましたね。

"CLS"には「画面をきれいにしてください」という意味があります。CLSのようにコンピュータが理解する言葉をキーワードといいます。キーワードは全部で200以上あります。いっぺんには覚えられませんから、少しずつ覚えるようにしましょう。

また、キーワードの最後に↵キーを押してください。これをしないとコンピュータは何もしてくれませんので忘れずに行ってください。

```
SHARP HuBASIC CZ-8FB02 Version1.0
Copyright (C) 1984 by SHARP/Hudson
Printer : CZ-800P
```

```
NEWON■
```

図1

```
SHARP HuBASIC CZ-8FB02 Version1.0
Copyright (C) 1984 by SHARP/Hudson
Printer : CZ-800P
74493 Bytes free
```

```
NEWON
Ok
■
```

図2

```

      .
      .
      .
Ok CLS■
```

図3

```
Ok
■
```

図4



それでは次に、**PRINT**というキーワードについてみてみましょう。

```
PRINT "X1"
```

と打って $\leftarrow$ キーを押して画面をみると図5のようになっていることがわかりますね。

```
PRINT "23+44"
```

と打って $\leftarrow$ キーを押すと、図6のように表示されます。

これは、**PRINT**が「引用符（"）で囲まれた文字を画面に表示しなさい」という意味をもったメッセージだからです。

```
PRINT "X1"
X1
Ok
■
```

図5

```
PRINT "23+44"
23+44
Ok
■
```

図6

それでは、次に示す例を順に試してください。 $\leftarrow$ の記号は「その位置で $\leftarrow$ の記号を打ってください」という意味です。

(例1) `PRINT 23+44`  $\leftarrow$

```
67
```

```
Ok
```

■

(例2) `PRINT (23+44)*2/5`  $\leftarrow$

```
26.8
```

```
Ok
```

■

(例3) `?23+44`  $\leftarrow$

```
67
```

```
Ok
```

■

上の例は(例1)が $23+44$ のたし算を、(例2)が $(23+44) \times 2 \div 5$ の計算をそれぞれコンピュータにさせ、その結果を表示させたものです。

これは、**PRINT**に「引用符（"）で囲まれた文字を表示しなさい」という意味以外に「計算した結果を表示しなさい」という意味もあるので、それを利用したものです。

なお、(例3)だけはちょっと別のキーワードの例のようですが、実は**PRINT**は?（疑問符）で置き換えて使うことができます。キーボードからP、R、I、N、Tと5文字入れるところが?の1文字で済むのでこれはたいへん便利ですね。

これまでみてきたように、キーワードは、その名の通り、あなたがコンピュータにメッセージを伝えるための鍵となる単語で、キーボードから正しく打ち込んで $\leftarrow$ キーを押すと、コンピュータはそのメッセージに応じた仕事をしてくれます。

## 2

## まずは簡単なプログラム作りから

それでは、

```
PLAY  $\leftarrow$ 
```

と打ち込んでください。

```
Missing operand
```

Ok



というメッセージが返って来ました。

これは、コンピュータが「PLAYというキーワードの意味はわかるけれども、PLAYの後ろに必要なデータが書かれていないので何もできません」と途方に暮れているのです。その必要なデータをいっしょに打ち込んでみましょう。

PLAY "BG"

さあ、どうになりましたか？何にも起こらないという人は本機の音量調整を上げてもう一度打ち込んでみてください。「シー・ソー」という電子音が出るはずです。

PLAYは音を出すためのキーワードで、音を表わす文字を引用符（"）で囲んで指定すると、音を出すことができます。ここでは、B（シの音）とG（ソの音）を指定したので「シー・ソー」という音が出たのです。これは、さきほどのPRINTと使い方が似ています。

PRINT "CHIME"

打ち込むと、引用符で囲んだ文字CHIMEが表示されますね。

では、次のように打ち込むとどうでしょう？

10 PRINT"CHIME"

何ごととも起こりませんね。「CHIME」と表示しないし、「Syntax error」などのメッセージが返って来るわけでもありません。

今度は、

20 PLAY"BG"

と打ち込んでください。やはり何ごととも起きません。でも、次のように打ち込むとどうなりますか？

RUN

画面に「CHIME」と表示されて、「シー・ソー」という音が出たでしょう。

RUNと打ち込んで、キーを押してやれば、何度でも同じことを繰り返すことができます。いったいどういうわけでしょう。

LIST

と打ち込んでみてください。

10 PRINT"CHIME"

20 PLAY"BG"

Ok



と画面に表示されるでしょう。

PRINTやPLAYの前に10とか20の番号をつけるかつけないかでコンピュータの反応が違ってきます。

番号をつけずに、キーワードを打ち込んでキーを押すと、コンピュータはすぐその場で実行して、画面に文字を表示したり、音を出したりします。

番号をつけると、コンピュータはすぐに実行しませんが、打ち込まれたメッセージをきちんと覚えているので、あとからRUN と打ち込めば何度でも実行することができます。ここで、

10 PRINT"CHIME"

20 PLAY"BG"

の2行のことをプログラムといい、10や20の番号のことを行番号、行番号の後ろのPRIN



T " CHIME " や P L A Y " B G " のことを ステートメント といいます。


また、プログラムを実行するために打ち込んだ R U N というキーワードのことを コマンド といいます。コマンドは、プログラムをどうするかコンピュータに指示するときに使います。アンダーラインのついた言葉は大切ですのでよく覚えるようにしてください。

R U N は「プログラムを実行しなさい」

L I S T は「プログラムを画面に表示しなさい」

C L S は、「画面に書かれた内容を消しなさい」


という意味のコマンドです。

L I S T 

と入れると、画面にプログラムが表示されます。

R U N 

と入れると、プログラムを実行します。


C L S 

と入れると、画面がきれいになりますが、プログラムもいっしょにきれいに消えてしまったかというと思ってはいけません。その証拠に、

L I S T 

と入れると、プログラムが表示されます。

## 3 プログラムの仕組み

コンピュータには、命令を入力すると、電卓のようにすぐ答えが表示される **直接実行** と、命令を入力しても R U N  を押さないかぎり答えを表示しない **間接実行** があります。一般に、間接実行のことを " プログラム " と呼んでいます。

### 3. 1 直接実行と間接実行のちがい


間接実行させるには行番号を各行のあたりに付けてコンピュータに順序よく仕事をさせるようにしますが、行番号を付けずに命令を入力するとすぐに実行され結果が表示されます。これが直接実行です。

たとえば  $4 + 5$  の計算をさせたいとき、右の上の行のようになるとすぐ答の 9 が下側に表示されますね。これが直接実行です。

```
PRINT 4+5
9
Ok
■
```

こんどは行番号を付けて右のように入力します。このときは画面に結果があらわれません。

```
10 PRINT 4+5
  (これを行番号と呼びます。)
RUN
9
Ok
■
```

ここで R U N  と入力するとはじめて答の 9 が表示されます。

これが間接実行と呼ばれるもので行番号を付けることによってその内容は「プログラム」として扱われすぐに実行されませんが、何度も繰り返し実行させたり、順序よく実行させることができます。

### 3. 2 プログラムって何だろう

間接実行によってコンピュータに仕事をさせようとする場合、「プログラム」が必要になります。

プログラムというのは、コンピュータにどういう順序で仕事をさせるかを書いた手引書のようなもので、コンピュータはプログラムに書かれている行番号の小さい順に仕事をこなして（実行して）いきます。

たとえばコンピュータに1から15までの数字を画面に表示させるプログラムを書いてみると次のようになります。（ここから $\boxed{\leftarrow}$ キーの表示は記していません。）

```
10 FOR I=1 TO 15
20 PRINT I
30 NEXT I
40 END
```

これがプログラムです。

```
10 FOR I=1 TO 15
20 PRINT I
30 NEXT I
40 END
RUN
1
2
3
4
:
15
OK
```

プログラムを実行させた画面

ここで、RUN（プログラムを実行せよ）と入力すると行番号10、20…の順にコンピュータは実行していきます。40番を実行してプログラムの実行が終了します。このようにプログラムは基本的には若い行番号から順に実行されます。

### 3.3 変数を使う

プログラムの中では、いろいろな数値を扱いますが、値の決まっていないものやいろいろと値が変わるものは**変数**という形で扱うと便利です。

たとえば次のプログラムは、たし算をさせようとするプログラムです。この中のAやB、Cが変数で、アルファベットまたはアルファベットと数値の組合せで名前を付けます。ただし変数の名前は240字以下で最初の1文字はアルファベットでなければなりません。

```
10 INPUT A,B
20 C=A+B
30 PRINT C
40 END
```

このプログラム中のAやB、Cが変数です。AやBはいろいろ値を変えることができます。CはA+Bの計算結果で値が変わりますのでこれも変数です。



一般に使用している数値にも整数や小数、分数などがあるようにコンピュータで使うことのできる変数にも、整数型、単精度実数型、倍精度実数型、文字型などの区別があり用途に応じて使い分けます。

### 3.4 困ったとき助けてくれる $\boxed{\text{CTRL}}$ + $\boxed{\text{D}}$ キー

本機を使用されて行く途中、次表のようなことで困ったことが生じるかも知れません。これらは無意識のうちにコンピュータをそのような状態にさせた結果ですから、次のような命令で正常にもどしてください。

$\boxed{\text{CTRL}}$  キーを押しながら  $\boxed{\text{D}}$  キーを押して、  
そのあと  
 $\boxed{\text{SHIFT}}$  キーを押しながら  $\boxed{\text{CLR HOME}}$  キーを押します。



1	キー入力された文字がすべて点滅文字になってもとにもどらないとき。
2	キー入力された文字がすべて反転文字になってもとにもどらないとき。
3	キーから入力された文字の内容が、キーの表示内容と合わなかったりわけのわからないパターンが出てきたとき。
4	キーから入力した文字が標準の大きさでなく、倍文字となるとき。
5	キーから入力した文字の色を白色にもどしたいとき。
6	 +  キーを押しても画面の文字が消えずに残っているとき。
7	4 0 文字モードで、キー入力しても画面に何もあらわれないとき。
8	コンピュータの音が鳴りっぱなしで止まらないとき。
9	画面のグラフィックがC L S 0 の命令でも消えないとき。
1 0	キー入力した文字がグラフィックのうしろにかくれてしまうとき。
1 1	グラフィックで描いた図形が指定した色と合わないとき。
1 2	L I S T, F I L E S コマンドなどによって、グラフィック画面が表示されなくなったとき。

## 4 プログラムの作成(その1)

### 4.1 プログラムの入力

プログラムをキーボードから打ち込んでコンピュータに覚え込ませることを、プログラムを**入力する**といいます。

さきほど入力したプログラムはたった2行でしたが、何十行、何百行となるのが普通です。そういった大きくて複雑なプログラムを入力する際には必ず途中で追加、変更、削除などの編集を行わねばなりません。

ここでは、小さなプログラムを例にとり、プログラムの編集方法を具体的に説明しましょう。

まず、NEWというコマンドを使って、コンピュータ内に残っているプログラムをきれいに消しましょう。

NEW 

試しに、

L I S T 

と打ち込んでみてください。

O k




と表示されて、プログラムがきれいに消えていることがわかります。

次に画面をきれいにしましょう。

C L S 

さあ、プログラムを入力する準備は整いました。スペルを間違えないように、プログラムを入力して行きましょう。

10 PRINT "X1" 

プログラムを1行入力したら、まず正しく入力されたかどうか見なおします。間違いが見つかったら、面倒でももう一度入力しなおしてください。何も全部入力しなおさなくても、もっとよい方

法があるのですが、それについてはあとで説明します。

正しく入力されたことを確かめて、

```
R U N ↵  
X 1  
O k  
■
```

と表示されたらオーケー。もし、

```
S y n t a x   e r r o r   i n   1 0  
O k  
■
```

とメッセージが出たら、

```
10 PRINT"X1" ↵
```

と再び入力しなおしましょう。

```
20 PLAY"BGADDR" ↵と入力し、さらに  
L I S T ↵と入力すれば
```

```
10 PRINT"X1"  
20 PLAY"BGADDR"  
O k  
■
```

と表示されます。

これでプログラムが2行になりました。実行してみましょう。

```
R U N ↵  
X 1
```

と表示されて音が出るでしょう。

```
30 PLAY"DE#FGGR" ↵  
L I S T ↵
```

```
10 PRINT"X1"  
20 PLAY"BGADDR"  
30 PLAY"DE#FGGR"  
O k  
■
```

これでプログラムが3行に増えました。R U N ↵と入力して、このプログラムを走らせる（R U Nの本来の意味は「走る」です）と、「X 1」と表示して、さきほどより長いメロディーが流れます。

## 4.2 1行の挿入

---

```
15 PRINT"CHIME" ↵  
L I S T ↵
```

```
10 PRINT"X1"  
15 PRINT"CHIME"  
20 PLAY"BGADDR"  
30 PLAY"DE#FGGR"  
O k  
■
```



10と20の間に新しい15の行が**挿入**されて、プログラムが4行に増えました。

```
RUN ↵  
X1  
CHIME
```

上のように表示されて、音が流れましたか？

プログラムは基本的には若い行番号から順に実行されます。ここでは、まず行番号10のPRINT、次いで15のPRINT、そして20と30のPLAYの順に実行されます。

## 4.3 1行の削除

---

それでは、X1と表示しないようにするにはどうすればよいのでしょうか。最も簡単な方法はプログラムから10番の行を取ってしまえばよいのです。そのためには、行番号のうしろに何も入力しないでリターンキーを押せばよいので、

```
10 ↵
```

と入力します。

```
LIST ↵  
  
15 PRINT"CHIME"  
20 PLAY"BGADDR"  
30 PLAY"DE#FGGR"  
Ok  
■
```

10番が削除されて3行のプログラムになっていることがわかります。走らせると、

```
RUN ↵  
CHIME
```

と、「X1」を表示しないでメロディーが流れます。

このように、プログラムから1行削除するには、その行番号だけを入力して↵キーを押せば済みます。

## 4.4 1行の変更

---

30のPLAYの音を変えてみましょう。30のステートメントを新たに入力すればよいのです。


```
30 PLAY"DABGGR" ↵  
LIST ↵  
  
15 PRINT"CHIME"  
20 PLAY"BGADDR"  
30 PLAY"DABGGR"  
Ok  
■
```

前の30番の行が、いま入力した内容に**変更**されました。さっそくプログラムを走らせてみましょう。どんな音が出るのかな？

```
RUN ↵
```

もし「Syntax error」が出たら、もう1度入力しなおすことを忘れずに。

```
5 CLS ↵
```

LIST 

```
5 CLS
15 PRINT"CHIME"
20 PLAY"BGADDR"
30 PLAY"DABGGR"
Ok
```

CLSは、「画面をきれいにする」という意味のコマンドですが、プログラムのステートメントとしても使うことができます。

RUN 

画面をきれいに消してから「CHIME」と表示して音がでるようになりました。

## 4.5 行番号のつけなおし

---

RENUM 

Ok

あれっ、何をしたのかな？

LIST 

```
10 CLS
20 PRINT"CHIME"
30 PLAY"BGADDR"
40 PLAY"DABGGR"
Ok
```




よく見るとプログラムの行番号が変わったのがわかります。でも、プログラムの内容まで変わったわけではないので安心してください。

RUN 

と入力して走らせると、コンピュータは、前と同じことをやってくれるでしょう。

RENUMは、「プログラムの行番号をつけなおしなさい」という意味をもつコマンドです。このコマンドは、プログラムに新しい行を挿入するために行間のスペースがなくなったときに使うと便利です。

さて、プログラムを増やしましょう。

```
50 PLAY"GBADDR" 
60 PLAY"DABGGR" 
70 PRINT"END" 
```

プログラムが正しく入力されたかどうかを確かめてから、走らせてみましょう。

LIST 

```
10 CLS
20 PRINT"CHIME"
30 PLAY"BGADDR"
40 PLAY"DABGGR"
50 PLAY"GBADDR"
60 PLAY"DABGGR"
70 PRINT"END"
```



O k

R U N 

どうですか。チャイムらしい音が出ましたか。


このプログラムはいま先頭の番号が100になっていますが、RENUMを使ってプログラムの先頭の行番号を変えることができます。たとえば、先頭を1000にしたいときは

RENUM 1000 

O k

■

と入力します。

L I S T 

```
100 CLS
110 PRINT"CHIME"
120 PLAY"BGAADR"
130 PLAY"DABGGR"
140 PLAY"GBADDR"
150 PLAY"DABGGR"
160 PRINT"END"
```

O k

■

試しに走らせてみると、さきほどと同じプログラムであることがわかるでしょう。

R U N 


## 4.6 複数行の削除

---


さて、今度はこのプログラムを消すことを考えてみましょう。残らず消してしまうにはNEWというコマンドを使えばよいのですが、もし100～160まであるプログラムのうち110～150の行を消したいときにはどうすればよいのでしょうか。

消したい行の行番号を1つ1つ入力して行くのもよいでしょうが、面倒だという人のためにもっとよい方法があります。それは、DELETEというコマンドを使う方法です。このコマンドを使えば、プログラムのいらなくなった行をまとめて消してしまふことができます。

110～150の行を消したいときは次のように入力します。

DELETE 110-150 

これで、コンピュータは110～150のプログラムをすべて消してしまいました。確かめるため、プログラムを見てみましょう。

L I S T 

```
100 CLS
160 PRINT"END"
```

O k

■

このプログラムを走らせると、画面が消えて「END」と表示されるはずです。

R U N 

E N D

O k

■

ここで、これまで説明したプログラムの編集方法について1度まとめておきましょう。

## ※ プログラムの編集方法(その1) ※

### 1. 1行挿入するには

⇒挿入したい行の上と下の行の行番号の間の番号を行番号に選ぶ。

たとえば、行番号20と30の間に挿入したいときは21～29の行番号で始まる1行を入力する。

### 2. 1行削除するには

⇒削除したい行の行番号のみを入力する。

### 3. 何行かまとめて削除するには

⇒DELETEコマンドを使う。

たとえば、行番号110～180を削除したいときは、

DELETE 110-180

と入力する。

### 4. 1行変更するには

⇒変更したい行の行番号で始まる1行を改めて入力する。

## 5 プログラムの作成(その2)

次に進む前に、プログラムと画面をきれいに消しておきましょう。

NEW

Ok

■

CLS

Ok

■

### 5.1 カーソルの移動と文字の修正





ここでは、新たに簡単な曲を演奏するプログラムを入力してみましょう。

10 PLEY"CRGRGRGRARARGRR"

ここでPLAYと入力するところを誤ってPLEYと入力したことに気づいたとします。

こういう場合、どうやって誤りを訂正したらよいでしょうか。もう1度「10…」と行番号10から入力しなおすのも一つの手ですが、もっとよい方法があります。

それは、キーボードのテンキー部分の下方にあるカーソルコントロールキーを使う方法です。

カーソルコントロールキーは矢印のついたキーで、の4つあり、画面上のカーソルを上下左右好きな方向に動かすことができます。



PLEYの「E」を「A」になおすためには、まずカーソルコントロールキーを使ってカーソルをPLEYの「E」のところまでもって行きます。

10 PLEY"CRGRGRGRARARGRR"

■

↑

カーソル

の状態で、を1回、を5回打ってみてください。カーソルが、ちょうど「E」のところに来たらオーケーです。もし、「E」を通り過ぎてしまっても、慌てないで4つのカーソル・キーをうまく使って「E」のところを持って行ってください。

10 PL[E]Y"CRGRGRGRARARGRR"

(□はカーソルを表わす)



ここでAを押すと、「E」が「A」に変わってPLAYとなります。カーソルは「Y」のところに來ていますから、そのままそこで $\leftarrow$ キーを押してください。

これで修正が完了しました。

試しにプログラムを見てみましょう。

```
LIST  $\leftarrow$ 
10 PLAY"CRGRGRGRARARGGR"
Ok
■
```

それでは次に、1行入力中、 $\leftarrow$ を押す前に入力の誤りに気づいた場合について見てみましょう。

```
20 PLEY ■
```

ここまで打って、PLAYの「A」が「E」になっている誤りに気づいたとき、それをなおすには2通りの方法があります。

まず、第1の方法は左向き矢印のカーソルコントロールキー $\leftarrow$ を2回打ってカーソルを「E」のところに持って行き、Aと打つ方法がありますが、第2の方法として $\boxed{\text{INS DEL}}$ キーを2回打って、

```
20 PL ■
```

「EY」の2文字を消してしまい、A、Yと打って修正する方法もあります。

さて、どちらかの方法で修正が終わったらプログラムの入力続けることにしましょう。

## 5.2 効率のよい入力方法

```
20 PLAY"FRFRERERDRDRCCRR"  $\leftarrow$ 
30 PLAY"GRGRFRFRERERDDRR"  $\leftarrow$ 
```

次の40番の行が上の30番とまったく同じステートメントでよいときは、カーソルを30の「3」のところに持って行って、4を打ち $\leftarrow$ キーを押します。

```
20 PLAY"FRFRERERDRDRCCRR"
40 PLAY"GRGRFRFRERERDDRR"
■
```

これで30番とまったく同じ行がもう1つ、40番の行に作られました。  
プログラムを見てみましょう。

```
LIST  $\leftarrow$ 
10 PLAY"CRGRGRGRARARGGR"
20 PLAY"FRFRERERDRDRCCRR"
30 PLAY"GRGRFRFRERERDDRR"
40 PLAY"GRGRFRFRERERDDRR"
Ok
■
```

どうです。40番の行がきちんとできているでしょう。

次の50番の行は10番の行とまったく同じでよいのですが、どうすればよいのでしょうか？  
そうです。10の「1」のところにカーソルを持って行って5と打って $\leftarrow$ キーを押せばよいのです。

```
50 PLAY"CRGRGRGRARARGGR"
20 PLAY"FRFRERERDRDRCCRR"
30 PLAY"GRGRFRFRERERDDRR"
40 PLAY"GRGRFRFRERERDDRR"
Ok
■
```

次の60番の行は20番の行とまったく同じです。いまちょうど、カーソルが20の「2」のところに來ていますね。その位置で6と打って $\leftarrow$ キーを押してください。

```

50 PLAY"CRGRGRGRARARGGR"
60 PLAY"FRFRERERDRDRCCRR"
30 PLAY"GRGRFRFRERERDDRR"
40 PLAY"GRGRFRFRERERDDRR"

```

Ok



プログラムの上にカーソルがあり、プログラムを書き換えてしまう恐れのあるときは、**CLR HOME** キーを使って画面を消してしまいましょう。

**SHIFT** キーを押しながら、**CLR HOME** キーを押してみてください。画面がすっかりきれいになりましたね。

ここで、プログラムを見てみましょう。

LIST

```

10 PLAY"CRGRGRGRARARGGR"
20 PLAY"FRFRERERDRDRCCRR"
30 PLAY"GRGRFRFRERERDDRR"
40 PLAY"GRGRFRFRERERDDRR"
50 PLAY"CRGRGRGRARARGGR"
60 PLAY"FRFRERERDRDRCCRR"

```

Ok



これは「キラキラ星」でだれもが知っている有名な曲ですが、このままこのプログラムを実行すると、テンポが非常に遅いながらも曲が流れます。

RUN

音楽を途中で止めたいときは、**SHIFT** キーを押しながら、**BREAK** キーを押してください。一般的に、実行中のプログラムを止めるには、**SHIFT** キーを押しながら、**BREAK** キーを押します。

曲のテンポを速くするには、曲のテンポを指定するステートメントを入れます。

```

5 PLAY300 

```

ついでに、画面を消すステートメントと、タイトルを表示するステートメントも入れましょう。

```

3 CLS 

```

```

4 PRINT"キラキラ星" 

```

LIST

```

3 CLS
4 PRINT"キラキラ星"
5 PLAY300
10 PLAY"CRGRGRGRARARGGR"
20 PLAY"FRFRERERDRDRCCRR"
30 PLAY"GRGRFRFRERERDDRR"
40 PLAY"GRGRFRFRERERDDRR"
50 PLAY"CRGRGRGRARARGGR"
60 PLAY"FRFRERERDRDRCCRR"

```

Ok





行番号を揃えて、

RENUM 

Ok

■

LIST 


```
10 CLS
20 LOCATE12, 5:PRINT"キラキラ星"
30 PLAY300
40 PLAY"CRCRCRGRARARGGR"
50 PLAY"FRFRERERDRDRCCRR"
60 PLAY"GRGRFRFRERERDDRR"
70 PLAY"GRGRFRFRERERDDRR"
80 PLAY"CRCRCRGRARARGGR"
90 PLAY"FRFRERERDRDRCCRR"
Ok
```

■

走らせてみると、「キラキラ星」が演奏されます。

## 5.3 文字の挿入

ここで、タイトルの「キラキラ星」を画面の中央に表示したいときは、タイトルを書く位置を **LOCATE** というキーワードを使って指定します。

「15 LOCATE12, 5 」と、15番の1行を入力してもよいのですが、20の番号とPRINTの間に「LOCATE12, 5:」を挿入しても同じプログラムになります。

まずカーソルをPRINTの「P」のところに持って行きましょう。そして、**SHIFT** + **INS DEL** キーを12回押す（**SHIFT** を押したまま **INS DEL** キーを12回打つ）と、カーソルのあるP以降が1文字ずつ計12文字分右にずれ、挿入するスペースがあきます。

20 **P** RINT"キラキラ星"

↑

カーソル


↓

20 ■

PRINT"キラキラ星"

↑


カーソル

ここで「LOCATE12, 5:」と入力して、キーを押すと挿入修正した20番の行が入力されます。

```
20 LOCATE12, 5:PRINT"キラキラ星"
30 .....
```

.....

プログラムを確かめてみましょう。

LIST 

```
10 CLS
20 LOCATE12, 5:PRINT"キラキラ星"
30 PLAY300
40 PLAY"CRCRCRGRARARGGR"
50 PLAY"FRFRERERDRDRCCRR"
60 PLAY"GRGRFRFRERERDDRR"
70 PLAY"GRGRFRFRERERDDRR"
80 PLAY"CRCRCRGRARARGGR"
90 PLAY"FRFRERERDRDRCCRR"
Ok
```

■

さっそく実行してみましょう。画面に「キラキラボシ」と表示されて音楽が流れます。



## 5.4 文字の削除

LIST 

```
10 CLS
20 LOCATE12, 5:PRINT"キラキラボシ"
30 PLAY300
40 PLAY"CRCRCRGRARARGGRR"
50 PLAY"FRFRERERDRDRCCRR"
60 PLAY"GRGRFRFRERERDDRR"
70 PLAY"GRGRFRFRERERDDRR"
80 PLAY"CRCRCRGRARARGGRR"
90 PLAY"FRFRERERDRDRCCRR"
```

Ok

■

「LOCATE12, 5:」を消したいときは、まず、カーソルを消したい文字のすぐ後ろ、ここではPRINTの「P」のところに持って行きます。そして、キーを12回打つと、カーソルの左隣の文字が1文字ずつ計12文字消され、それに続く文字の「PRINT…」を引っ張ってきます。ここでキーを押すと、削除修正した行が入力されます。

```
20 LOCATE12, 5:PPRINT"キラキラボシ"
```

↓

↑



カーソル


```
20 PRINT"キラキラボシ"
```

```
30 .....
```

↑

カーソル

キーを押しながら キーを押して画面を消してから、プログラムを確かめましょう。

LIST 

```
10 CLS
20 PRINT"キラキラボシ"
30 PLAY300
40 PLAY"CRCRCRGRARARGGRR"
50 PLAY"FRFRERERDRDRCCRR"
60 PLAY"GRGRFRFRERERDDRR"
70 PLAY"GRGRFRFRERERDDRR"
80 PLAY"CRCRCRGRARARGGRR"
90 PLAY"FRFRERERDRDRCCRR"
```

Ok

■

これまで簡単な例を示しながら、プログラムの編集についてほんの触りの部分を説明しました。これまでプログラムを作るということが、どのようなことなのか、ある程度おわかりになられたと思います。



### 1. カーソルはカーソルキーを使って自由に移動できる。

- ⬆カーソルを上へ移動する。
- ⬇カーソルを下へ移動する。
- ⬅カーソルを左へ移動する。
- ➡カーソルを右へ移動する。


### 2. 文字を挿入するには

⇒カーソルを削除したい部分のすぐ後に持って行って、**SHIFT** + **INS DEL** キーを必要なだけ押す。

### 3. 文字を削除するには







⇒カーソルを削除したい部分のすぐ後に持って行って **INS DEL** キーを必要なだけ押す。

### 4. 画面をきれいにするには

⇒ **SHIFT** + **CLR HOME** キーを押す。(= C L S )  
 ( **SHIFT** + **CLR HOME** は「**SHIFT** キーを押しながら **CLR HOME** キーを押す」の意味です)

## ◇ コマンドのまとめ ◇

いままで出てきた基本的なコマンドについてまとめてみましょう。

- |   |                        |
|---|------------------------|
| 1. C L S                 | 画面をきれいにする。             |
| 2. N E W                 | 記録されたプログラムを消す。         |
| 3. L I S T               | 記録されているプログラムを画面に表示する。  |
| 4. R U N                 | プログラムを実行する。            |
| 5. R E N U M             | プログラムの行番号をつけなおす。       |
| 6. D E L E T E    n — m  | プログラムの n 番から m 番までを消す。 |

## 絵を描く、色を塗る

まず、キーボードより、

INIT ↵

WIDTH 40, 25, 0, 1 ↵

と入力してください。(INIT, WIDTHについては応用編の「ディスプレイモード」で詳しく説明します。)

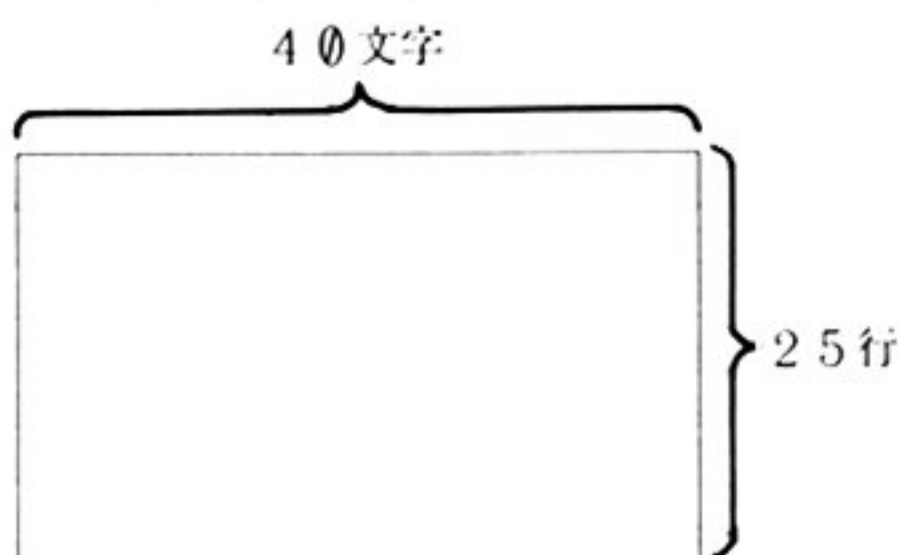
このように指定すると、

表示文字数                    40文字×25行(1,000文字)

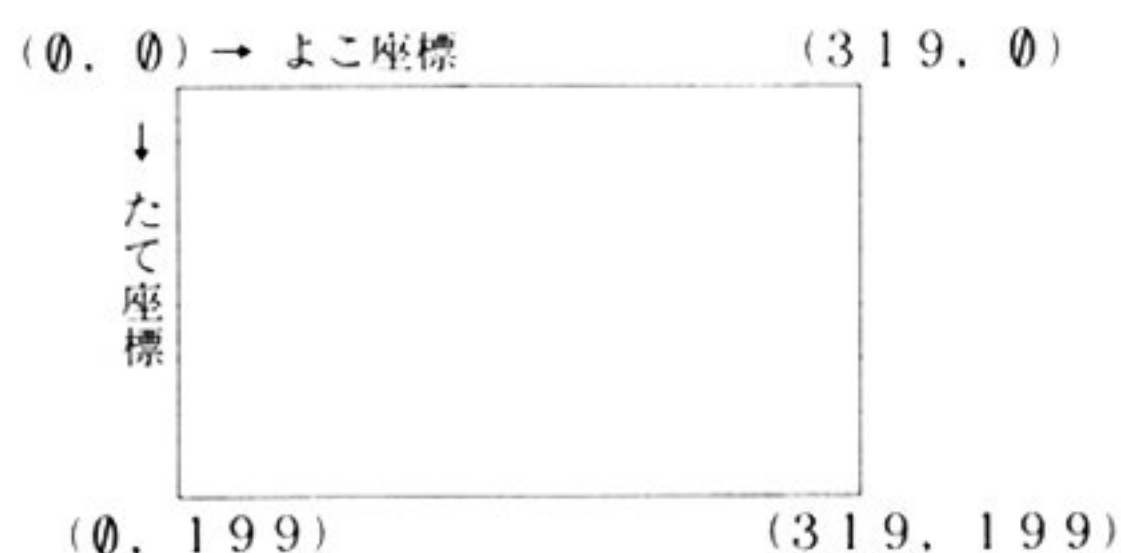
グラフィック解像度        320×200ドット

の画面設定になります。

●文字を表示する場合



●図形(グラフィック)を描く場合



本章では、この画面を使用します。

(なお、本機ではWIDTHステートメントにより各種の画面設定ができます。)  
(詳しくは、応用編「ディスプレイモード」を参照してください。)

## 1 点をうつ

次のように入力してください。

NEW ↵

10 WIDTH 40: INIT ...横40文字×縦25行の設定。

20 SCREEN 0 ...グラフィック画面の使用モードの指定。

30 CLS ...画面をきれいにします。

40 PSET(160,100) ...白い点を座標(160, 100)に表示。

各行ごとに必ず↵キー(リターンキー)を入力してください。そして

RUN ↵

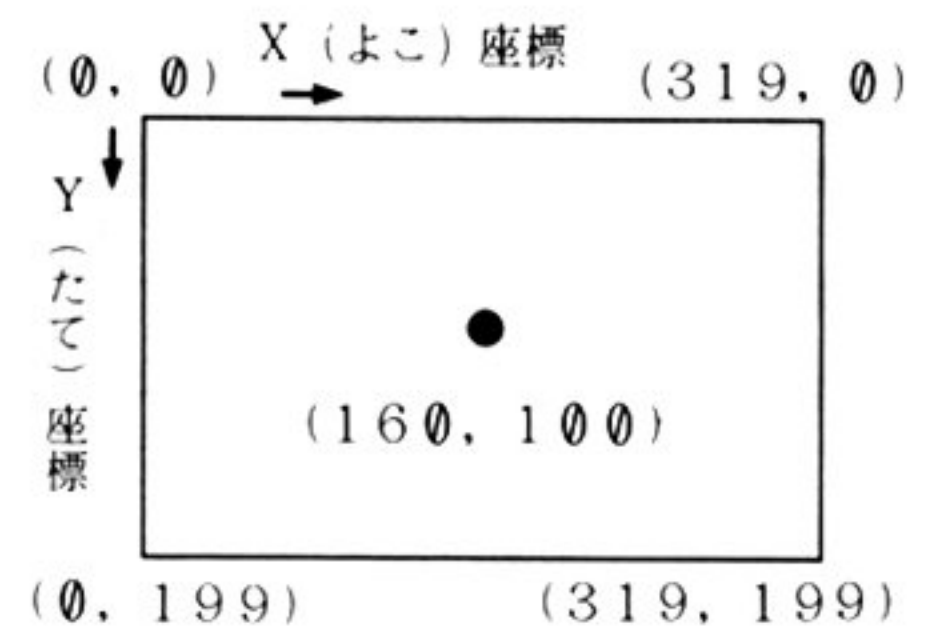
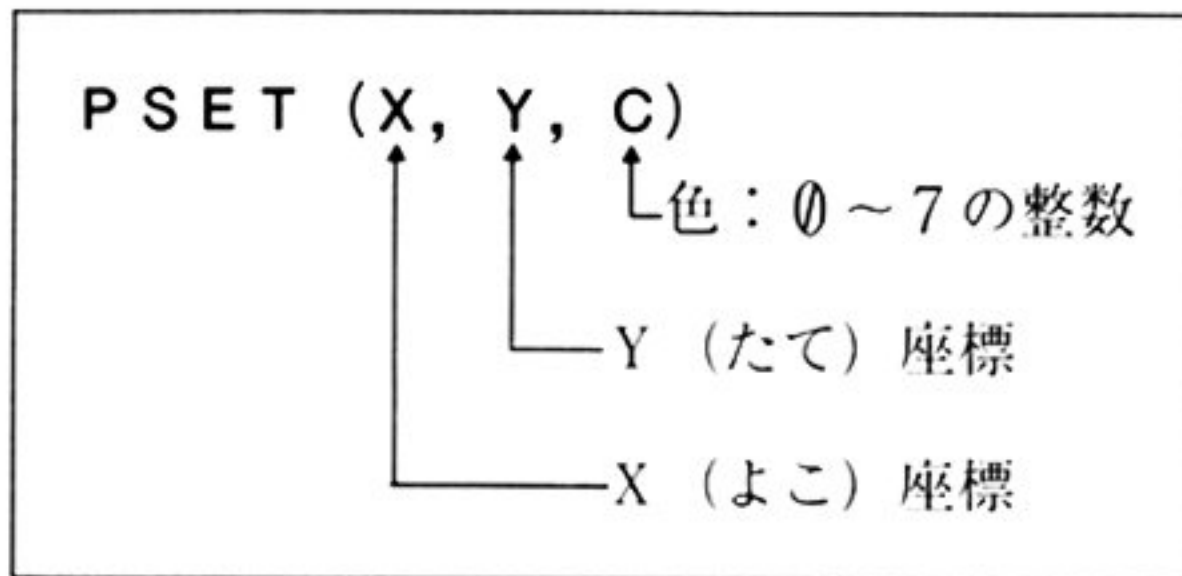
と入力して実行すると、画面のほぼ中央に白い点が表示されます。



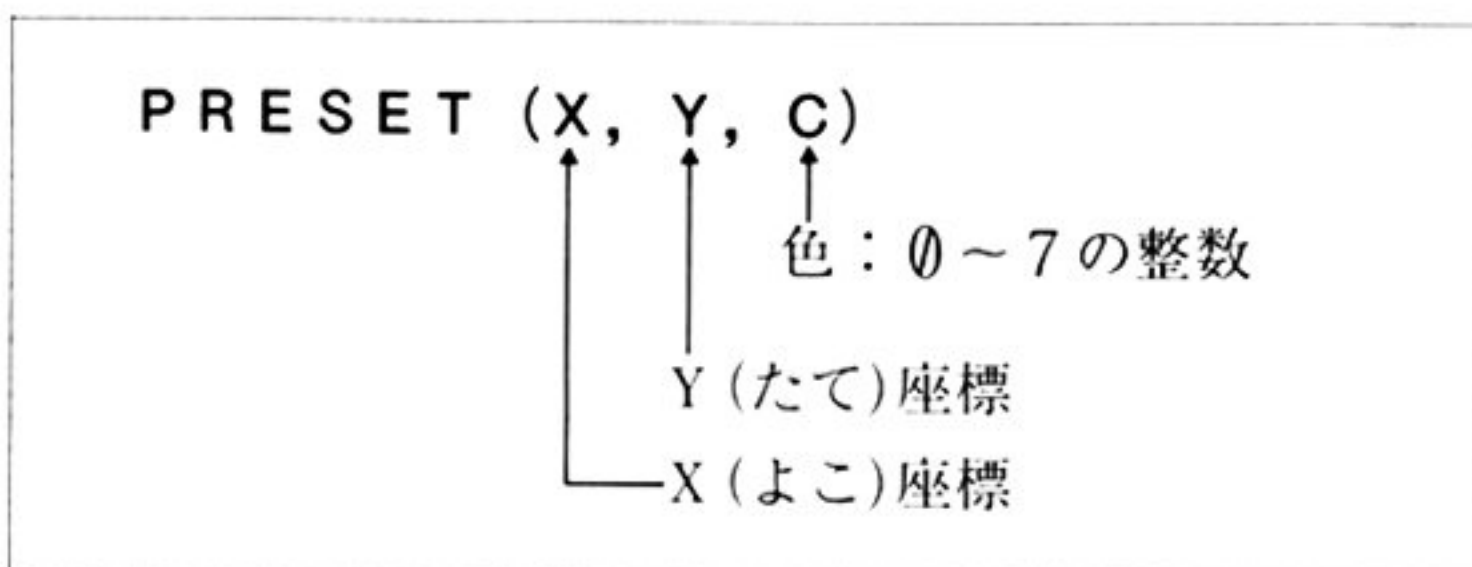
40行を

**PSET (160, 100, 2)**

と変えてみると先ほどと同じ位置に赤色の点が表示されます。



なお、このように、**PSET**ステートメントは点を表示するのに対し、**PRESET**ステートメントは逆に点を消す働きをします。



正確に言えば、**PRESET**は点を消すのではなく、指定した色を消す命令です。たとえば、白色の点に対して**PRESET**で緑色を消すようにすると、その点の色はマゼンタに変わります。これは青、赤、緑の3色からなる白色から緑色を消した場合、残るのは青、赤からなるマゼンタだからです。

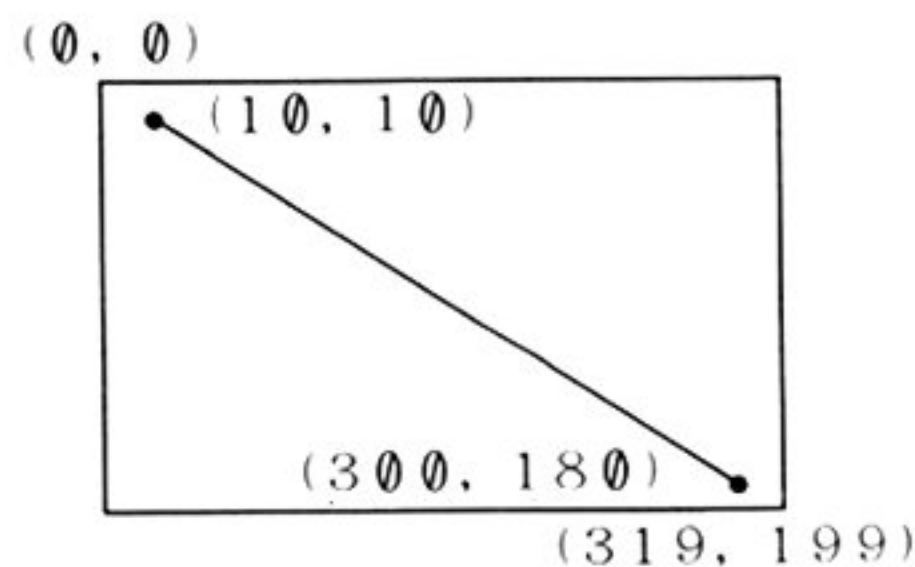
## 2

## 直線を描く

線を描くには、LINE命令を使います。  
たとえば、NEWと入力し、

```
10 WIDTH 40:INIT
20 SCREEN0,0
30 CLS4
40 LINE(10,10)-(300,180),PSET,4
```

次にRUNと実行すると画面の左上から右下へ緑色の直線が描けます。



LINE (X1, Y1) - (X2, Y2), PSET, C

引き始めの座標

引き終わりの座標

色の番号：0～7の整数

PSET文

### サンプル プログラム

#### コンピュータグラフィック花火

画面上に定めた点と不特定な点とを線で結ぶように描かせるものです。

```
10 REM *** LINE ***
20 WIDTH 40:INIT
30 CLS 4
40 FOR I=1 TO 300
50 X=INT(RND*320)
60 Y=INT(RND*200)
70 C=INT(RND*7)+1
80 LINE(160,100)-(X,Y),PSET,C
90 NEXT
100 END
```

Xに0から319までの整数をランダムに入れる

Yに0から199までの

Cに1から7までの

(160, 100)から50, 60, 70行で決められた点に線を描く

LINE命令は、他にもいろいろな機能を持っています。詳しくは応用編「グラフィックス」をお読みください。

色の指定の次にB (Box) をつけると、指定した2つの点を対角線とする長方形を描くことができます。たとえば

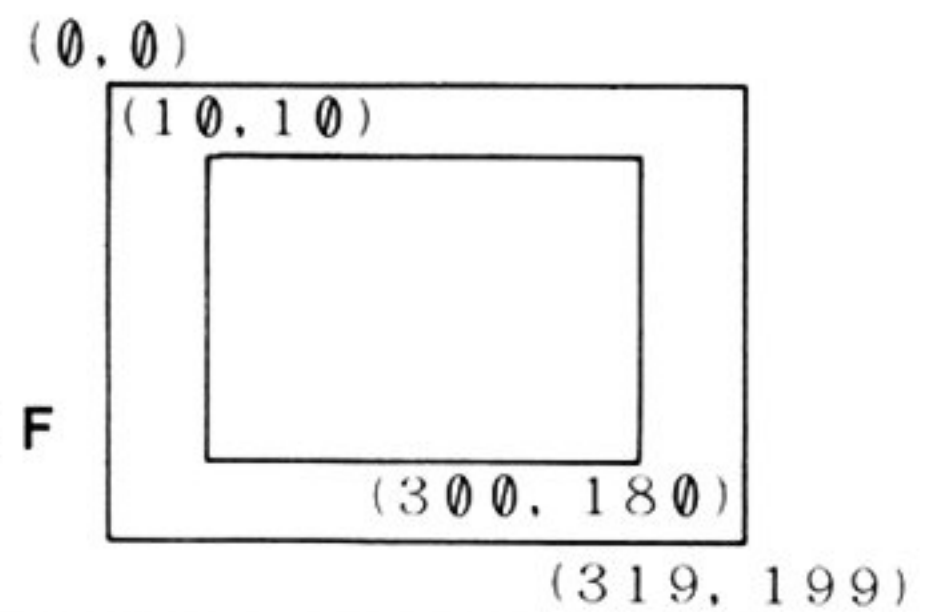
LINE (10, 10) - (300, 180), PSET, 4, B



とすれば、緑色の長方形が描けます。

さらに、BのかわりにBF (Box Fill) とすると、長方形を描いてその内側を塗りつぶすことができます。

LINE (10, 10) - (300, 180), PSET, 4, BF



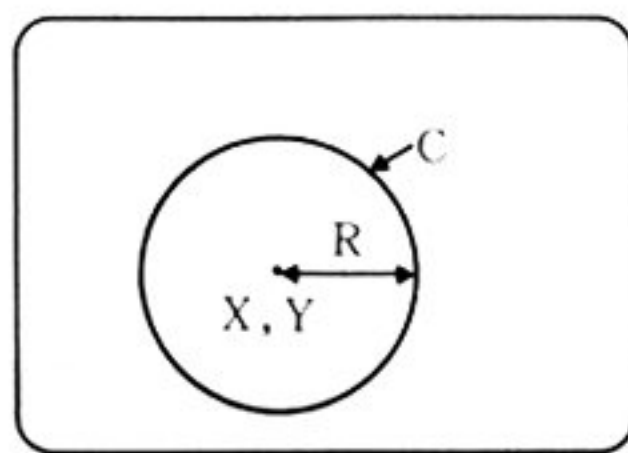
### 3 円を描く

今度は円を描いてみましょう。円を描くにはCIRCLE命令を使います。

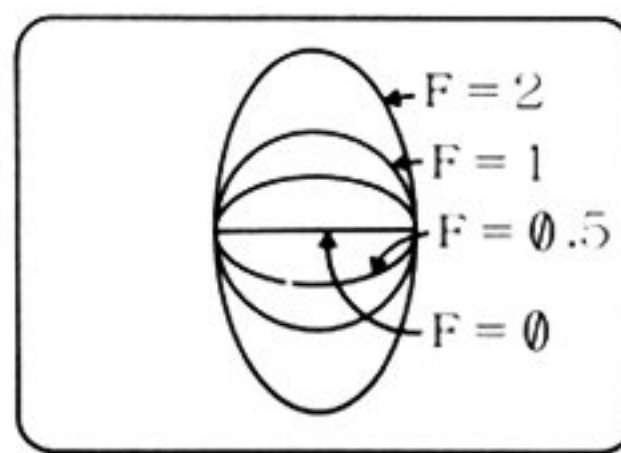
CIRCLE (X, Y), R, C, F,  $\theta_s$ ,  $\theta_e$

↑                    ↑                    ↑                    ↑                    ↑  
 中心座標                    半径                    色                    偏平率                    初期角                    終了角

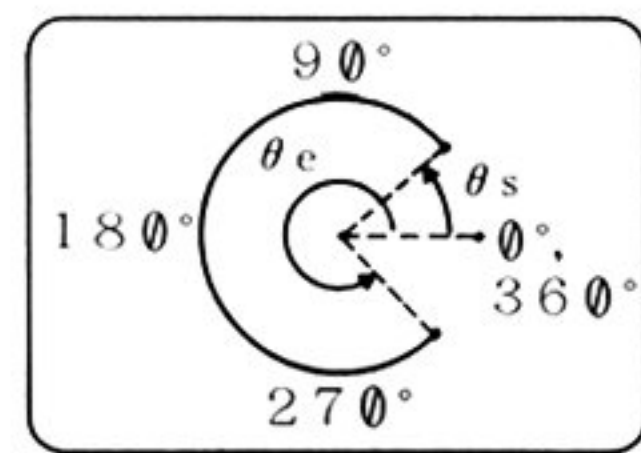
上式でC, F,  $\theta_s$ ,  $\theta_e$ は省略することができます。偏平率Fを1に設定すると、正円になりそれ以外は楕円になります。



円の中心座標 X, Y  
 半径 R  
 円を描かせる色 C



偏平率 F  
 F=2  
 F=1  
 F=0.5  
 F=0 } の参考図



初期角  $\theta_s$   
 終了角  $\theta_e$   
 $\theta_s=45$   
 $\theta_e=315$  } の参考図

WIDTH 40: SCREEN 0, 0: CLS 4: CIRCLE (160, 100), 50, 2 と入力してください。赤の円が描けますね。

次のサンプルプログラムを入力してください。カラーコードの1から7までを繰り返しながら偏平率と半径を変化させて円を描きます。

#### サンプルプログラム

##### カラーサークル

カラーコードの1から7までを繰り返しながら縦横比と半径を変化させて円を描きます。

```
10 REM *** CIRCLE ***
20 WIDTH 40: INIT
30 CLS 4
40 R=1
50 FOR C=1 TO 7
60 CIRCLE (160, 100), R, C, N
70 N=N+.05
80 IF R>95 THEN END
90 R=R+5
100 NEXT C
110 GOTO 50
```

← 半径Rの初期値を1に設定  
 ← 円の色を変化させる  
 ← 縦横比Nを0.05ずつ加算  
 ← 半径が95を越えるとEND  
 ← 半径Rを5ずつ加算

## 4

## 多角形を描く

次は正多角形を描きましょう。正多角形はPOLY命令を使います。  
書式としては次のとおりです。

POLY	(X, Y),	R,	C,	$\Delta\theta$ ,	$\theta_s$ ,	$\theta_e$
	↑	↑	↑	↑	↑	↑
	中心の座標		色		初期角	
		↑		↑		↑
	中心から頂点までの距離		ステップ角		終了角	

正n角形を描くには次のように設定します。

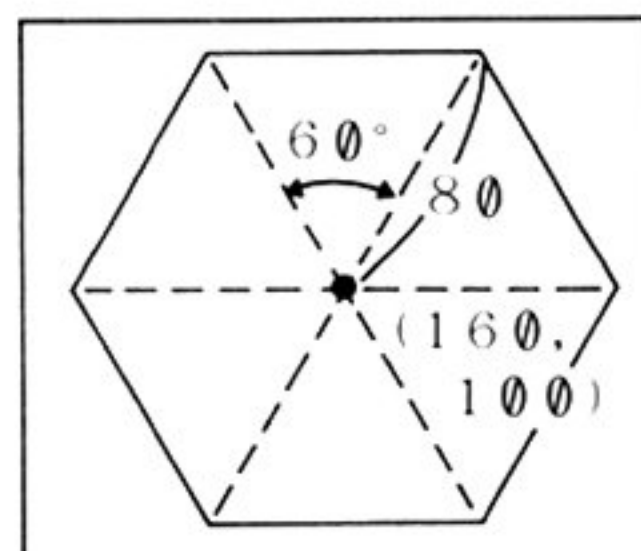
$$\Delta\theta = 360 / n$$

$$\theta_e = \theta_s + 360$$

たとえば、正六角形を描いてみます。正六角形のステップ角は60度（ $360^\circ \div 6$ ）です。

```
10 WIDTH 40:SCREEN0,0:INIT
20 CLS4
30 POLY(160,100),80,2,60
```

これで、中心座標（160, 100）、頂点距離80の赤い正六角形が描かれました。



## 5

## 色を塗る

線で描いた図形に色を塗ってみましょう。色を塗るにはPAINT命令を使います。書式としては次のとおりです。

PAINT	(X, Y),	C,	B,	$B_1, B_2, \dots B_6$
	↑	↑		↑
	塗り始める座標	塗る色		境界色

PAINT命令は（X, Y）で指定された座標から塗りつぶしを開始し、境界色で指定された色で囲まれた範囲を塗りつぶします。境界色は何色かにまたがる場合がありますので、1色だけでなく何色でも指定できます。

では実際に色を塗ってみます。

```
10 WIDTH 40:INIT
20 CLS4
30 CIRCLE(200,100),80,2
40 PAINT(200,100),1,2
```

これを実行すると、中心座標（200, 100）、半径80の赤円内を青色で塗りつぶします。



## サンプル プログラム

プログラムは、同心円をいくつか描き、円と円で囲まれたドーナツ状の部分を違った色で塗りつぶすものです。

```
10 REM **** PAINT ****
20 WIDTH 40:INIT
30 CLS 4
40 FOR R=1 TO 99 STEP 12 }
50 CIRCLE(160,100),R,7 } ← 半径Rが1から99までの同心円を9個描く
60 NEXT R
70 Y=5
80 FOR C=1 TO 7 }
90 PAINT(160,Y),C,7 } ← 同心円の内側をカラーコード1から7までの色で順に塗っていく
100 Y=Y+12
110 NEXT C
```

# 3章

## プログラムの保存と再生

### 3章

パーソナルコンピュータはプログラムを入力させたり、またそれを実行させたりしますが、これはパーソナルコンピュータの内部にあるIC（集積回路）が処理しています。このICは、いくつものプログラムを入れたり出したりする必要があるため、コンピュータの電源を切ると憶え込んでいたプログラムがすべて消滅するようにつくられています。

ですから、プログラムを保存するには、別の記憶場所へ移さなくてはなりません。

これをプログラムのSAVE（セーブ）といいます。ここではフロッピーディスクとカセットテープへのSAVEおよびLOAD（再生、ロード）について説明します。

なお、カセットを使用する場合は、オプションの専用データレコーダが必要です。

### 1 フロッピーディスクに保存、再生する場合

#### ■SAVE……

- ①プログラムを保存するにはSAVE命令を使います。

たとえば、次のプログラムをセーブしようとしています。

```
10 INPUT "A=";A
20 INPUT "B=";B
30 C=A+B
40 PRINT "A+B=";C
50 END
■←カーソル
```

- ②フォーマット<sup>1)</sup>され、ライトプロテクト<sup>2)</sup>のかかっていないフロッピーディスクを本体にセットします。

- ③適当なプログラム名（ただし13字以内）をつけます。上記のプログラム名を「タシザンノプログラム」として、次のように入力します。

SAVE"0:タシザンノプログラム"



保存するデバイス名 (フロッピーディスク) (0番目)      保存するプログラムの名前

- ④フロッピーディスクドライブのインジケータが点灯し書き込みを始めます。

- ⑤画面にOkと表示すれば、プログラムはフロッピーディスクにセーブされたことになります。

1) フォーマット ……フロッピーディスクを初期設定して、使用可能な状態にすることです。  
(フォーマットの仕方については『アプリケーションソフトの説明書』の「ディスクユーティリティ」を参照してください。)




2) ライトプロテクト…フロッピーディスクへの書き込み禁止。

ライトプロテクトノッチに保護用紙をはっておくと書き込みはできなくなります。

#### ■LOAD……

①フロッピーディスクから「タシザンノプログラム」を再生するには次のように入力します。

LOAD"0:タシザンノプログラム" 

再生するプログラムの名前


## 2

## カセットテープに保存、再生する場合

#### ■SAVE……

たとえば、前述の「タシザンノプログラム」というプログラムを保存する例を述べます。

- ①前述の「タシザンノプログラム」を入力します。
- ②ツメの折っていないカセットテープを専用データレコーダにセットし、テープカウンターを"000"としておきます。
- ③

SAVE"CAS:タシザンノプログラム" 

プログラムを保存するデバイス名（カセット）      保存するプログラムの名前

と入力します。

④専用カセットデータレコーダが自動的に作動を開始します。

⑤画面には

Writing"タシザンノプログラム、"

と表示が出、数秒後にカセットデータレコーダは自動的に停止し、

WRITEの表示ランプは消灯して画面にOkと出ます。

⑥これで「タシザンノプログラム」はカセットテープにセーブできました。


#### ■LOAD?……

以上で、プログラムのセーブは終了したわけですが、万一うまくセーブされてなかったら…

…と考えますね。せっかくのプログラムがセーブされていなければ困りますので

LOAD?という命令で確実にセーブできているか確認します。

①カセットテープをテープカウンター"000"まで巻戻して、セーブした「タシザンノプログラム」が確実にセーブできているか確かめるため次のように入力します。

LOAD?"CAS:タシザンノプログラム" 

確かめるプログラムの名前

- ②これを実行すると画面には次のようなメッセージが表示され、セーブされていれば数秒後に Ok が表示されます。

Found "タシザンノプログラム."

Ok




- ③もし、カセットテープなどに問題があってうまくセーブされていないときは次のメッセージが出ますので、カセットを交換してもう一度セーブをし直してください。

Tape read Error

#### LOAD.....

- ①プログラムをカセットテープから再生するときは、次のように入力してください。

LOAD"CAS:タシザンノプログラム" 

再生するプログラムの名前

- ②数秒後、画面に次のようなメッセージが出ます。

Found "タシザンノプログラム."

そしてカセットデータレコーダの動作がとまり、Ok の表示が出て、ロードが完了したことを知らせます。

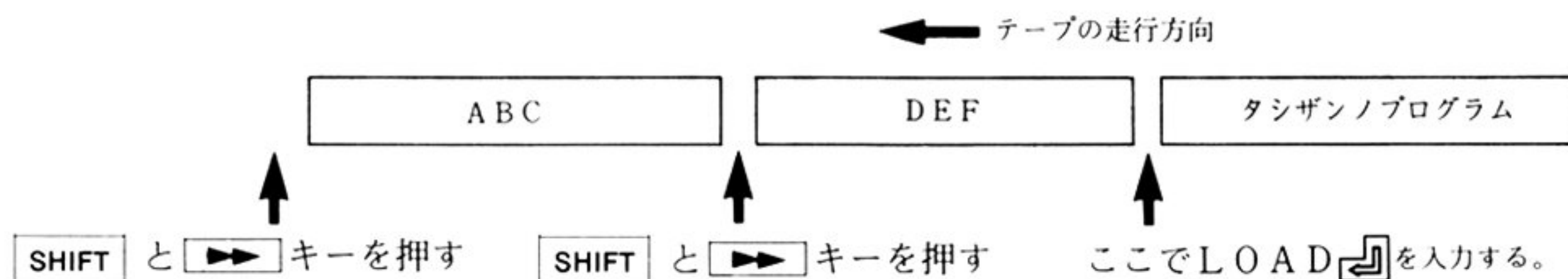
- ③なお、1 本のカセットテープに何種類かのプログラムが入っていてそのうしろへ先程の「タシザンノプログラム」をセーブした場合はロードするとき最初に説明したように「プログラム名」と一緒に入力します。すると次のようにロードする目的のプログラムに到達するまで関係のないプログラムはスキップしてゆきます。

LOAD"タシザンノプログラム"	:	→	"ABC" という名前のプログラム
Skip "ABC"	:	→	
Skip "DEF"	:	→	
Found "タシザンノプログラム"	:	→	"DEF" という名前のプログラム
Ok	:		

また、上の例で「タシザンノプログラム」は3 番目に入っていることがわかっていれば **SHIFT** キーを押しながらテープオペレーションキーを押すことによって次のプログラムの頭出しができますからこれを2 回繰り返すことで「タシザンノプログラム」の頭の部分へすぐに到達することができます。

(次の図を参照してください)





### 3 フロッピーディスクまたはカセットテープの内容を見るには

プログラムを記録したフロッピーディスクまたはカセットテープの内容を見るときは **FILES** という命令を使います。

この命令を実行すれば、プログラムの名前(ファイル名)だけを拾い出し、その一覧表を画面上に表示します。

#### ■フロッピーディスクの場合

プログラムのはいったフロッピーディスクをディスクドライブ0番にセットし、

**FILES "0:"** [enter]

と入力します。

(ディスク BASIC を使用したとき、初期状態ではファンクションキー **F1** を押せば **FILES "0:"** [enter] を実行します。)

#### ■カセットテープの場合

プログラムのはいったカセットテープをデータレコーダにセットし、テープを巻戻してから

**FILES "CAS:"** [enter]

と入力します。

なお、プリンタにファイル名を出力するには、フロッピーディスク、カセットテープそれぞれに対して

**LFILES "0:"** [enter]

**LFILES "CAS:"** [enter]

と入力します。

# 4章

## テレビをコントロールする

### 4章

本機では専用ディスプレイテレビを接続したときにキーボードやBASICの命令で自由にテレビをコントロールできます。

さらに本機にはカレンダー付タイマー機能がついていますから、これらを組み合わせて、専用ディスプレイテレビをつけたり、消したりのテレビコントロールが可能になります。ここではあらかじめ内蔵されているテレビタイマーコントロールを使ったものと、BASICの命令を使ったものの2通りの説明をしています。

ここで使うBASICの命令は次のとおりです。

ASK、TIME\$、DATE\$、DAY\$、TVPW、CRT、CHANNEL、  
VOL、SCROLL

### 1 テレビタイマーコントロール(TV Timer Control)

#### 1. 1 テレビタイマーコントロールの呼びだし

クロックおよびタイマーの呼び出しは、BASICを読み込んであるかないかによって操作が異なってきます。

##### ●BASICを読みこんでいない場合

専用ディスプレイテレビの画面上に次のメッセージが表示されますので、**[T]**キーを押してください。

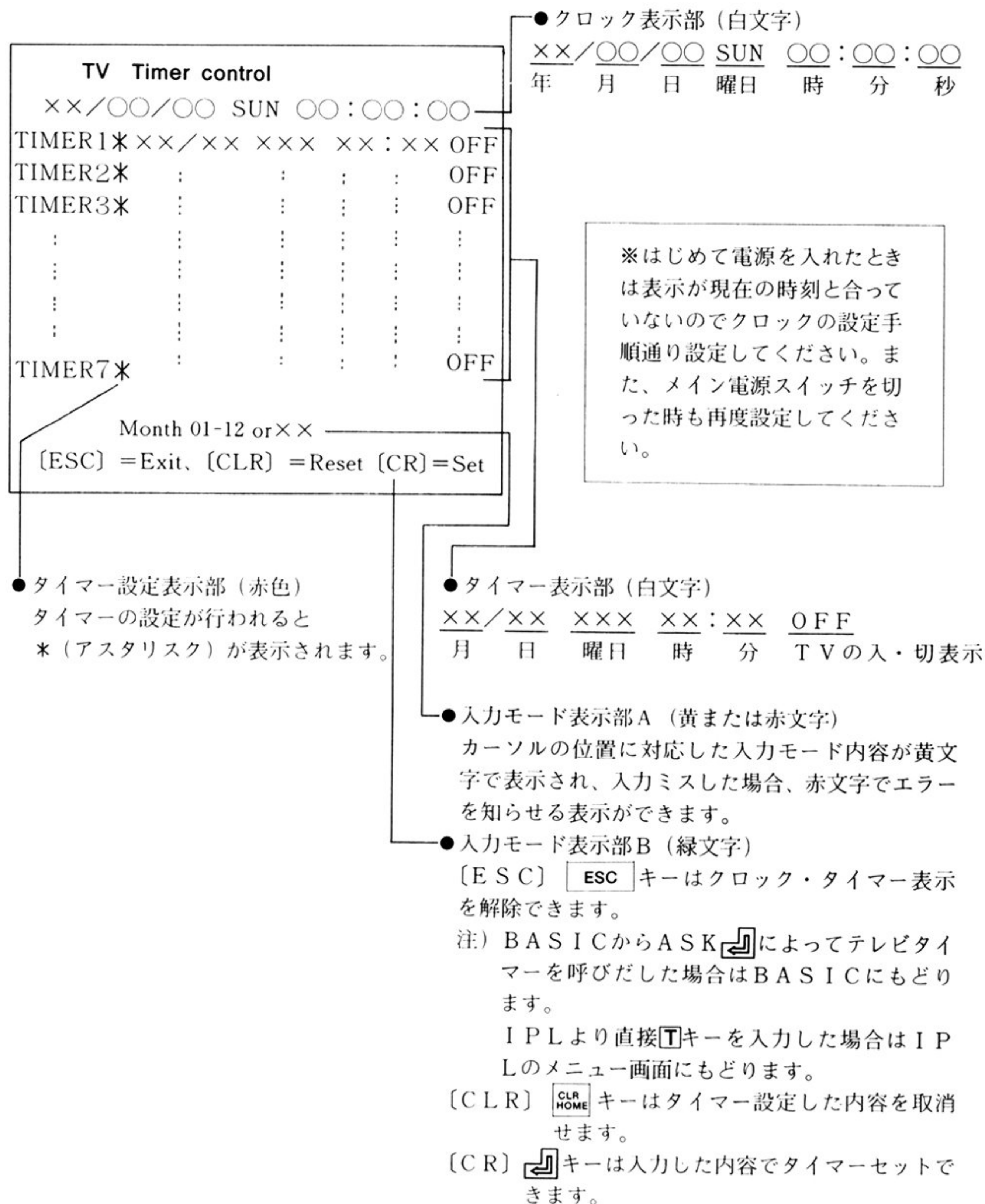
```
Make your device ready
Press selected key to start driving:
F:Floppy
R:ROM
C:CMT
T:Timer
```

●BASICを読み込ませてある場合はASK を入力してください。



## 1.2 テレビタイマーコントロールの表示内容

画面は下図のようにかわり、カーソルがT I M E R   ××／××…の最初の×のところで点滅しています。



### 1.3 クロック（時計）を現在時刻に合わせる

■現在時刻にコンピュータ内蔵のクロックを合わせる必要があるときは次の手順に従って設定してください。

ここでは1986年1月12日日曜日午後3時30分20秒と仮定して説明します。

1. カーソルをカーソルコントロールキーでクロック

表示部の左端に移動させます。

入力モード表示内容

2. 西暦1986年の末尾2桁[8][6]をキー入力します。

Year 00-99

3. 1月の[0][1]をキー入力します。

Month 01-12 or××

4. 12日の[1][2]をキー入力します。

Day 01-31 or××

5. 日曜日の[S][U][N]をキー入力します。

SUN MON THE WED THU FRI SAT or××

6. 午後3時（15時）の[1][5]をキー入力します。

Hour 00-23 or××

時刻は24時間表示ですから午後3時は15時になります。

Minute 00-59

Second 00-59

7. 午後3時30分の分の単位[3][0]をキー入力します。

8. 午後3時30分20秒の秒の単位[2][0]をキー入力します。

9. 設定時刻の3時30分20秒になった瞬間に[Enter]キーを押すとその時点からクロックはカウントをはじめ、クロックの設定は終了しカーソルはTIMER 1の頭のところへ移動します。

### 1.4 タイマーで番組予約をする。

クロックが動作を始めたところで今度は専用ディスプレイテレビの番組予約をするためのタイマー設定を行ってみましょう。

例1 ここでは、先程クロックを設定した翌日つまり1986年1月13日、月曜の午前9時30分からはじまる5チャンネルの1時間番組を番組予約すると仮定して説明します。

1. カーソルはタイマー表示部のTIMER 1 ××…

の最初の×のところにきているはずですが、そうでないときは最初の×のところへカーソルコントロールキーで移動します。

入力モード表示内容

Month 01-12 or××

2. 1月の[0][1]をキー入力します。

Day 01-31 or××

3. 13日の[1][3]をキー入力します。

SUN MON THE WED THU FRI SAT or××

4. 月曜日の[M][O][N]をキー入力します。

Hour 00-23 or××

5. 午前9時の[0][9]をキー入力します。

Minute 00-59

6. 午前9時30分の分の単位[3][0]をキー入力します。

TV Power ON? (Y or N)

7. カーソルはOFFのOのところで点滅しています。

TV Channel 1-12

この例では専用ディスプレイテレビをONさせたいわけですから入力モード表示にこたえるかたちで[Y]をキー入力します。

8. OFFに変わり表示された"ON CH"のうしろでカーソルが点滅していますので予約するチャンネルの[5]をキー入力します。

最後に[Enter]キー入力するとこの行のタイマーがセットされたことを示す赤色の\*（アスタリス



ク) がT I M E R 1のあとに表示されカーソルは次の行の頭へ移ります。

T I M E R 1 \* 1 / 1 3 S U N 0 9 : 3 0 O N C H 5

このようにタイマーセットがなされるとコンピュータ本体インジケータ部のT I M E R表示ランプが点灯し、タイマーが動作中であることを表示します。

## 1.5 タイマーでスイッチをOFFにする

こんどは専用ディスプレイテレビを指定された時刻にOFFさせるタイマーセットするため、さきほどの手順と同じように

T I M E R 2 \* 0 1 / 1 3 S U N 1 0 : 3 0 O F F

と入力してもよいのですが「1時間後にOFFさせる」のなら次の方法が簡単です。ただし、この場合毎日午前10:30には専用ディスプレイテレビをOFFするようにタイマーが働きますので、ご注意ください。

入力モード表示内容



Month 01-12 or × ×

TV Poewr ON? (Y or N)

9. カーソルはT I M E R 2 × × …の最初の×のところにきています。

10. カーソルキーでカーソルを順次右へ送り、午前10時30分の部分のみ入力します。

T I M E R 2 × × / × × × × × 1 0 : 3 0 O F F

11. **[N]** キーまたは、**[↵]** キーの入力でセットされカーソルは次の行の頭に移ります。

T I M E R 2 × × / × × × × × 1 0 : 3 0 O F F

これで専用ディスプレイテレビを指定された時刻にタイマーセットできました。

応用例

例2 毎週火曜日の午前11時から午後2時まで専用ディスプレイテレビをONさせ、10チャンネルを番組予約するときのタイマー表示がどうなっているかをごらんください。

T I M E R 3 \* × × / × × T U E 1 1 : 0 0 O N C H 1 0

T I M E R 4 \* × × / × × T U E 1 4 : 0 0 O F F

■例1～2の内容を見て、お気付きのように、このタイマーは×（つまり内容を規定しない）を使用することで、プログラマーの思いのままに専用ディスプレイテレビをタイマーコントロールすることができるわけです。

■一度セットされたタイマーの内容は一度メイン電源を切るかまたはクリア（タイマーの取消し）をしない限り、働き続けますので、毎日繰り返すようなタイマー設定の場合は便利です。

## 1.6 タイマーを取消す

先程セットしたタイマー内容を取消す場合は次の手順で行います。T I M E R 1からT I M E R 6を取消すことを例にして説明します。

1. カーソルをT I M E R 1の行へ移動させます。この場合カーソルはT I M E R 1の行のどの位置でもかまいません。

2. **[SHIFT]** キーを押しながら **[CLR HOME]** 押すとT I M E R 1の内容は取消され表示はタイマー設定前の状態にもどります。

3. カーソルはT I M E R 2の行に移っていますので同様に **[SHIFT]** を押しながら **[CLR HOME]** キーを押し



T I M E R 2 を取消します。

同じ操作で T I M E R 6 まで全部取消すとコンピュータ本体の T I M E R 表示は消えタイマー設定されていないことを示します。

#### 〈ご注意〉

クロック表示とメイン電源の入・切について

- ・はじめてメイン電源を入れたときは、表示が現在時刻とちがっていますが、これは故障ではありません。クロックの設定手順を参照して、正確な時刻に合わせてください。
- ・また、メイン電源を切った状態では、クロック機能が働かなくなりますので、再設定が必要です。
- ・本機のクロックは、年号の自動切換え表示を行ないませんので、年度が改まるごとに新しい年号を設定してください。
- ・うるう年（2月29日のある年）にあたる場合は、月／日の再設定が必要です。
- ・メイン電源を切りますと年号は××に変わりますので再設定してください。

## 2 プログラムでテレビをコントロールする


- 次のステートメントで内蔵タイマーの設定・表示が行なえます。

T I M E \$ = " 時間 : 分 : 秒 "

D A T E \$ = " 年 / 月 / 日 "


D A Y \$ = " 曜日 "

次のように入力することで設定・表示が行なわれます。

T I M E \$ = " 1 2 : 3 4 : 5 6 " 


O k

■

D A T E \$ = " 8 5 / 1 2 / 0 7 " 


O k

■

D A Y \$ = " F R I " 

O k

■

P R I N T T I M E \$ , D A T E \$ , D A Y \$ 

12:36:03 85/12/07 FRI

Ok



- BASICでテレビをコントロールする命令はつぎのとおりです。

TVPW	{	ON	テレビの電源をつけます
		OFF	テレビの電源を消します

CRT	{	0	テレビ放送を表示します
		1	コンピュータ画面を表示します
		2	テレビ放送のコントラストを下げてコンピュータ画面を重ねて表示します
		3	テレビ放送とコンピュータ画面を重ねて表示します

\* SHIFT + 田の状態はCRT 2と同じです。

CHANNEL 1~12 チャンネルを設定します

VOL -62 (小さい) ~64 (大きい) 音量を変化させます

SCROLL N スーパーインポーズ画面のとき(CRT 2、CRT 3)、コンピュータ画面の上下方向のスクロールを行ないます。  
Nの値が正のときは上方向に、負のときは下方向にスクロールします。Nの値が0、省略したときストップします。  
ただし N=-3 から 3 までの整数

ではBASICプログラムでテレビをコントロールしてみます。次のプログラムを入力してみてください。



[例 1]

```
10 TVPW OFF  
20 IF TIME$ <> "07:00:00" THEN 20  
30 TVPW ON:CHANNEL 1
```

朝7時になるとテレビの電源が入って1チャンネルになります。20行の時刻、30行のチャンネルを変えることで好きな時刻に好きなチャンネルをつけることができます。これだけではテレビタイマーコントロールで設定するのと同じですから、テレビタイマーコントロールではできない機能を使って少し工夫したプログラムにしてみます。

[例 2]

```
10 INIT:WIDTH 40,25,0,1:CLS4  
20 CRT0:CHANNEL 5  
30 CRT 2  
40 CSIZE3:LOCATE 0,0:PRINT#0 TIME$  
50 GOTO 40
```

20行で指定したチャンネルに時間を大きく表示するようにしたプログラムです。

# 5章

## 漢字を表示する

本機には、強力な日本語処理機能を備えた漢字BASIC（CZ-8FB02）が標準装備されています。この章では、この漢字BASICを利用して、漢字、ひらがな等を表示してみます。（詳細は応用編「日本語処理」を参照してください。）

### 1 日本語の入力

BASIC・CZ-8FB02をロードしてください。（すでにBASICをロードしている場合も、念のためIPLのスイッチを押すなどしてBASICをロードし直してください。）

日本語は、日本語入力モードに入ってから入力します。次のようにキー入力してください。

(a) **CTRL** + **XFER** ( **CTRL** キーを押しながら **XFER** キーを押す)

または

(b) **SHIFT** + **XFER** ( **SHIFT** キーを押しながら **XFER** キーを押す)

日本語入力モードに入ると、表示画面の最下行に変換フィールドが現われます。

テキストエリア

変換フィールドエリア

なお、日本語入力モードから通常の状態に戻るには、再び前記(a)または(b)を実行します。

#### (1) 漢字の入力

では、実際に日本語を入力してみましょう。

(例)「日本」と入力するには…

① **N** **I** **C** **H** **I** とキー入力します。

■

[平仮名]にち■

← 全/半角 間/直接 ローマ字 音訓

- ② **XFER** キーを押して漢字に変換します。このとき、"にち"という読みを持つ漢字は、音訓辞書中には"日"しかありませんので、漢字グループを表示せずにテキストエリア上へ漢字"日"を表示します。

H■

[平仮名]■

← 全/半角 間/直接 ローマ字 音訓

- ③ 次に、**H O X** をキー入力します。（**X** キーは「ん」に対応しています。詳しくは後述の「カナ-ローマ字変換一覧表」をご覧ください。）

H■

[平仮名]ほん■

← 全/半角 間/直接 ローマ字 音訓



- ④ **XFER** キーを押して漢字変換します。すると、"ほん"の読みを持つ漢字グループがあらわれます。

H ■

[平仮名]ほん ■

本 翻 叛 奔 反 品

- ⑤ 漢字グループの中の該当文字"本"を選びます。⏏キー（リターンキー）を押すか、テンキー **1** を入力します。

日本 ■

[平仮名] ■

← 全/半角 間/直接 ローマ字 音 訓

以上で、漢字"日本"が入力されました。

次に、**HTAB** キーを使って、"日本"を表示してみましょう。

- ① **N I C H I** とキー入力します。

■

[平仮名]にち ■

← 全/半角 間/直接 ローマ字 音 訓

- ② **HTAB** キーを押します。すると記号 " <sup>^</sup>i " が表示されます。

H■

[平仮名]にち <sup>^</sup>i■

← 全/半角 間/直接 ローマ字 音訓

- ③ 続けて **H** **O** **X** とキー入力します。

■

[平仮名]にち <sup>^</sup>ほん■

← 全/半角 間/直接 ローマ字 音訓

- ④ **XFER** キーを押します。すると、まず最初の " にち " が " 日 " に変換されます。

H■

[平仮名]ほん■

← 全/半角 間/直接 ローマ字 音訓

⑤ 続けて、**XFER** キーを押すと、次の "ほん" の読みをもつ漢字グループがあらわれます。

H■	
[平仮名]ほん■	<b>本</b> 翻叛 奔反品

⑥ 漢字グループの中の該当文字 "本" を選びます。

日本■	
[平仮名]■	← <b>全/半角</b> <b>間/直接</b> <b>ローマ字</b> <b>音訓</b>

以上で漢字 "日本" が入力されました。

このように **HTAB** キーを用いると、1 度に漢字数文字分の読みを変換フィールドに入力でき、次々に漢字変換していくことができます。この時、**HTAB** キーは漢字の読みと読みの間に入力します。後は、**XFER** キーを押して漢字変換をくり返します。

変換フィールドに入力できるかな数は、全角文字で 20 字、半角文字で 40 字です。

**HTAB** 記号 " ^1 " は半角文字として入力されますので、1 度に変換できる漢字数はこれらより求めることができます。


- 1) 全角文字…漢字やひらがなのように 16×16 ドットで構成された文字
- 2) 半角文字…8×8 ドットまたは 16×8 ドットで構成された文字


ここで、漢字入力のしかたをまとめます。

①変換フィールド内に出力されたひらがなまたはカタカナは、**XFER** キーを押すことによってその読みに対応した漢字に変換されます。

漢字は画面右下に最高 9 文字ずつ系列されます。





②目的の漢字が見つからない場合は、**XFER** キー、スペースキー、またはカーソルコントロールキー  を押します。次の漢字グループが表示されます。


また、前のグループに戻すには、カーソルコントロールキー  を押します。

③目的の漢字が見つければ、その漢字を取り出します。その方法には次の2通りがあります。

1. **1** ～ **9** のテンキーを使って指定する。

漢字グループの左端の漢字が **1**、右端の漢字が **9** のキーに対応していますので、目的の漢字に対応したテンキーを押します。

2. カーソルコントロールキー 、 を使って指定する。

カーソルコントロールキーを使って目的の漢字の上にカーソルを合わせ、リターンキー  を押します。


④以上の操作によって、指定された漢字がテキストエリア内のカーソルの位置に表示されます。

⑤なお、漢字グループを表示した状態で、入力誤りに気づいたり、目的の漢字がなくて変換を中止したい場合は

**ESC** キー

を押します。すると、漢字グループの表示が消え、変換フィールド内にカーソルが戻りますので、ひらがなまたはカタカナの訂正ができます。なお、変換フィールド内で文字を入力しているとき、間違って入力した場合は

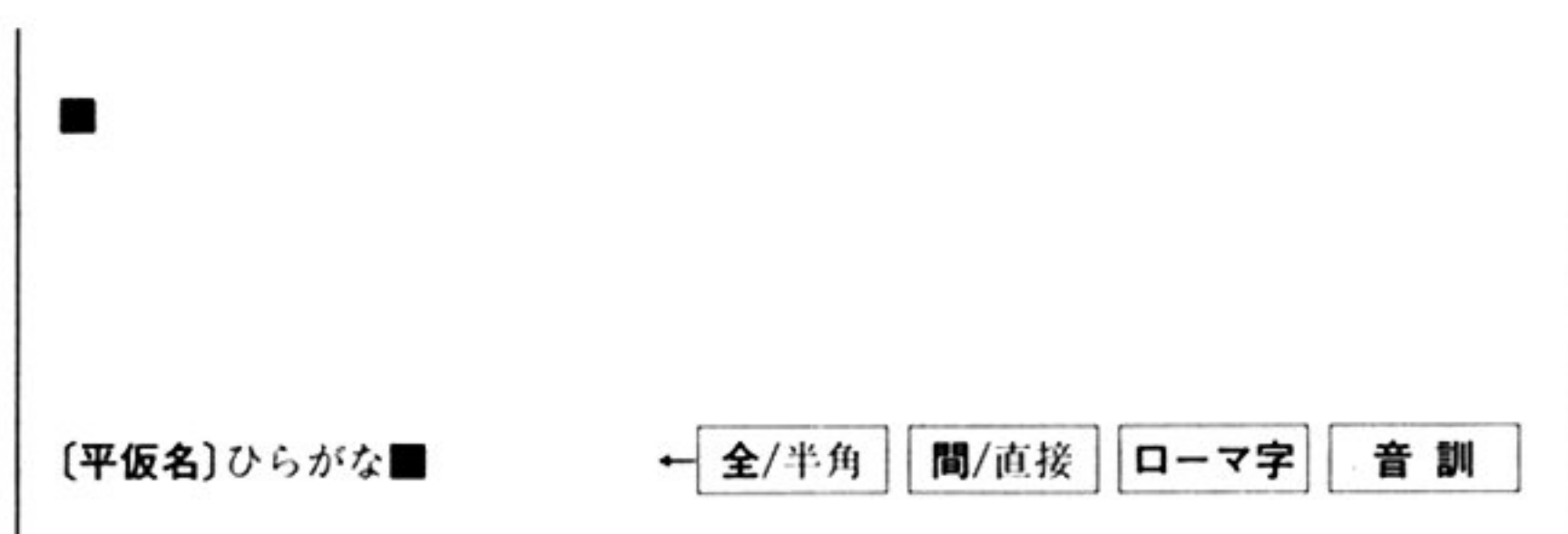
**INS DEL** キー

を押して入力し直してください。カーソルコントロールキー  を使用しても働きません。


## (2) ひらがなの入力

(例)「ひらがな」と入力するには…

① **H I R A G A N A** とキー入力します。



The screenshot shows a text input area with a cursor at the end of the text "HIRAGANA". Below the input area, there is a label "[平仮名] ひらがな" followed by a cursor. To the right of the label are four buttons: "全/半角", "間/直接", "ローマ字", and "音訓".

② リターンキー () を押せばテキストエリアに " ひらがな " を表示します。



The screenshot shows the same text input area as before, but now the text "HIRAGANA" has been replaced by "ひらがな". The label below the input area is "[平仮名]" followed by a cursor. The four buttons "全/半角", "間/直接", "ローマ字", and "音訓" are still present to the right of the label.

### (3) カタカナ、英数字の入力（全角文字）

(例1) 「カタカナ」と入力するには…

- ① ファンクションキー **F3** を押すと、〔平仮名〕が〔片仮名〕に変わります。

The screenshot shows a text input area with a cursor. Below it, the label reads 「片仮名」 followed by a small black square. To the right of the label are four buttons: 「全/半角」, 「間/直接」, 「ローマ字」, and 「音訓」. The label 「片仮名」 is positioned to the left of the 「全/半角」 button.

- ② **K A T A K A N A** とキー入力します。

The screenshot shows the text input area now containing the Katakana characters 「カタカナ」 followed by a small black square. The label below the input area is 「片仮名」カタカナ followed by a small black square. The buttons remain the same: 「全/半角」, 「間/直接」, 「ローマ字」, and 「音訓」.

- ③ リターンキー (↵) を押せばテキストエリアに "カタカナ" を表示します。

The screenshot shows the text input area with the Katakana characters 「カタカナ」 followed by a small black square. The label below the input area is 「片仮名」 followed by a small black square. The buttons remain the same: 「全/半角」, 「間/直接」, 「ローマ字」, and 「音訓」.

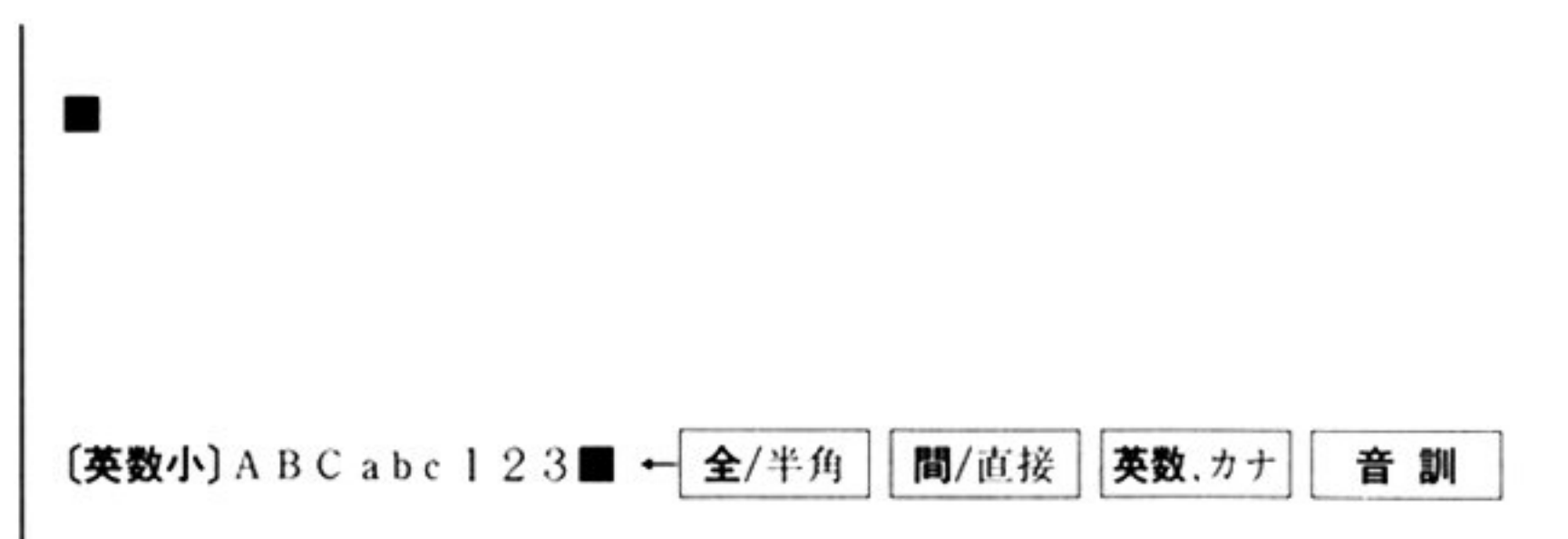
(例2) 「A B C a b c 1 2 3」と入力するには…

- ① ファンクションキー **F4** を押すと、下の画面になります。

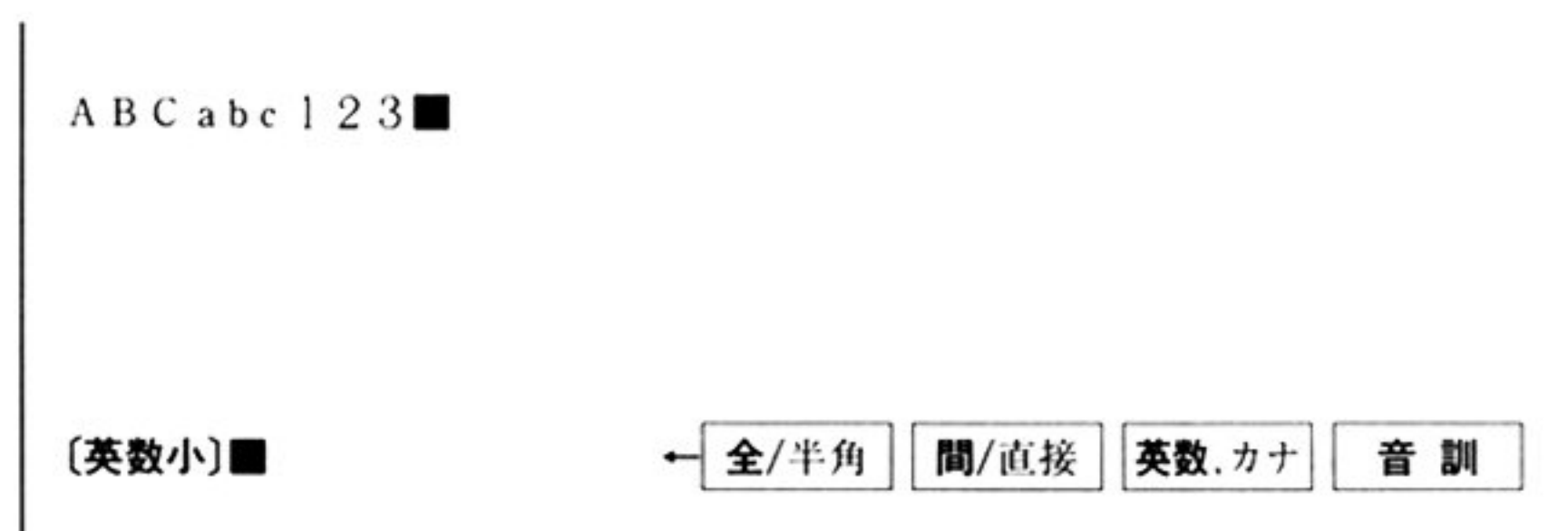
The screenshot shows a text input area with a cursor. Below it, the label reads 「英数大」 followed by a small black square. To the right of the label are four buttons: 「全/半角」, 「間/直接」, 「英数,カナ」, and 「音訓」. The label 「英数大」 is positioned to the left of the 「全/半角」 button.

キャピタルロックキー **caps lock** をロック状態にすると「英数大」になり、ロックを解除すると「英数小」になります。

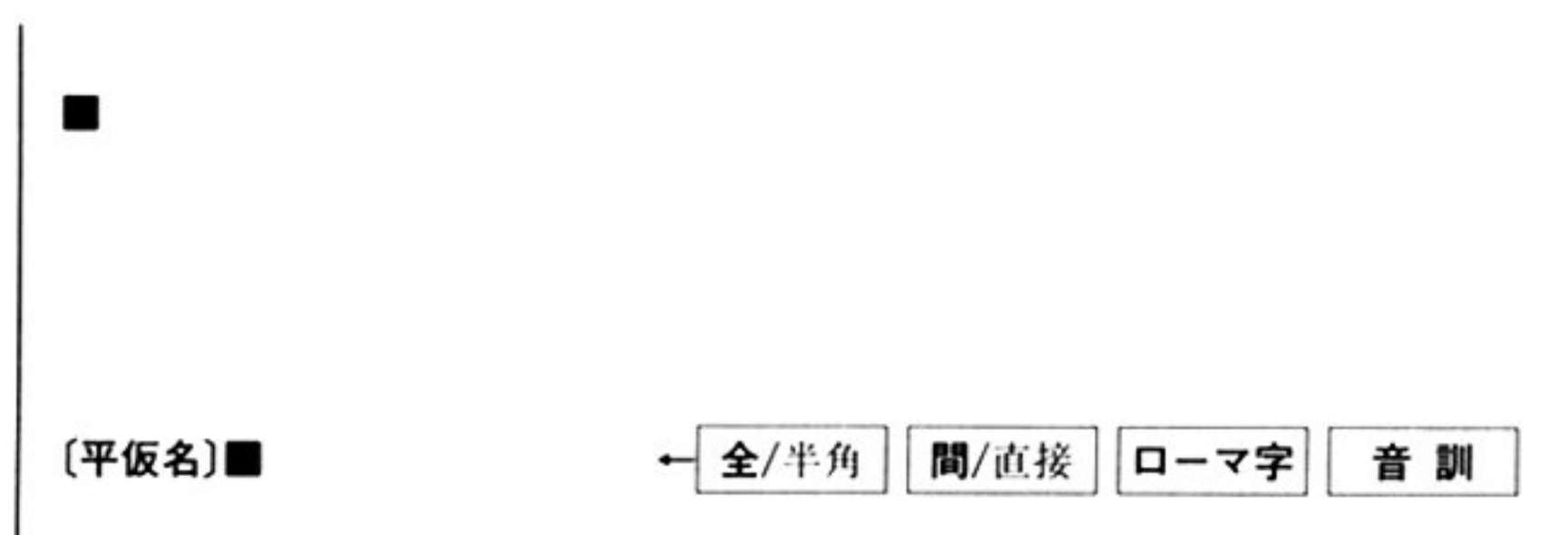
- ② [英数大] で **A** **B** **C** とキー入力し、**caps lock** キーのロックを解除して  
 [英数小] で **A** **B** **C** とキー入力し、さらにテンキーの **1** **2** **3** とキー入力します。



- ③ リターンキー (**↵** キー) を押せば、テキストエリアに " A B C a b c 1 2 3 " を表示します。



なお、この状態で次に漢字、ひらがなを表示するには、**F5** キー、続いて **F3** キーを押してください。次の画面になり、漢字、ひらがなが入力できます。



ここで、ファンクションキー **F3** ~ **F5** をまとめます。詳しくは応用編「日本語処理」をお読みください。

- F3** : 平仮名／片仮名…ひらがなまたはカタカナを出力するかを選択します。
- F4** : 英数字直接方式…キーボード上のアルファベット、数字、記号の通りに入力します。
- F5** : ローマ字-カナ変換方式…アルファベットを入力して、ひらがなまたはカタカナに変換します。



ローマ字-カナ変換一覧表

あ	い	う	え	お			
A	I	U	E	O			
か	き	く	け	こ	きゃ	きゅ	きょ
KA	KI	KU	KE	KO	KYA	KYU	KYO
さ	し	す	せ	そ	しゃ	しゅ	しゅ
SA	SI	SU	SE	SO	SYA	SYU	SYO
	SHI				SHA	SHU	SHO
た	ち	つ	て	と	ちゃ	ちゅ	ちょ
TA	TI	TU	TE	TO	TYA	TYU	TYO
	CHI	TSU			CHA	CHU	CHO
な	に	ぬ	ね	の	にゃ	にゅ	にょ
NA	NI	NU	NE	NO	NYA	NYU	NYO
は	ひ	ふ	へ	ほ	ひゃ	ひゅ	ひょ
HA	HI	HU	HE	HO	HYA	HYU	HYO
		FU					
ま	み	む	め	も	みゃ	みゅ	みょ
MA	MI	MU	ME	MO	MYA	MYU	MYO
や		ゆ		よ			
YA		YU		YO			
ら	り	る	れ	ろ	りゃ	りゅ	りょ
RA	RI	RU	RE	RO	RYA	RYU	RYO
わ		を		ん			
WA		WO		X			
が	ぎ	ぐ	げ	ご	ぎゃ	ぎゅ	ぎょ
GA	GI	GU	GE	GO	GYA	GYU	GYO
ざ	じ	ず	ぜ	ぞ	じゃ	じゅ	じょ
ZA	ZI	ZU	ZE	ZO	ZYA	ZYU	ZYO
	J I				JA	JU	JO
だ	ぢ	づ	で	ど	ぢゃ	ぢゅ	ぢょ
DA	DI	DU	DE	DO	DYA	DYU	DYO

ば	び	ぶ	べ	ぼ	びゃ	びゅ	びょ
BA	BI	BU	BE	BO	BYA	BYU	BYO
ぱ	ぴ	ぷ	ぺ	ぽ	ぴゃ	ぴゅ	ぴょ
PA	PI	PU	PE	PO	PYA	PYU	PYO
ふぁ	ふぃ		ふぇ	ふぉ			
FA	FI		FE	FO			

- ① 小文字は、**SHIFT** キーを押しながらアルファベットを入力することによって変換できます。

「あ」	.....	<b>SHIFT</b>	+	<b>A</b>
「い」	.....	<b>SHIFT</b>	+	<b>I</b>
「う」	.....	<b>SHIFT</b>	+	<b>U</b>
「え」	.....	<b>SHIFT</b>	+	<b>E</b>
「お」	.....	<b>SHIFT</b>	+	<b>O</b>
「ゃ」	.....	<b>SHIFT</b>	+	<b>Y A</b>
「ゅ」	.....	<b>SHIFT</b>	+	<b>Y U</b>
「ょ」	.....	<b>SHIFT</b>	+	<b>Y O</b>
「っ」	.....	<b>SHIFT</b>	+	<b>Z</b> または <b>SHIFT</b> + <b>T U</b> または <b>SHIFT</b> + <b>T S U</b> または <b>SHIFT</b> + <b>ろっ</b>

小文字の「っ」は、上記入力他に、同じアルファベットを2度連続して入力することによって表示させることもできます。

〔例〕ローマ字入力「**I**、**R**、**A**、**S**、**S**、**H**、**A**、**I**」→カナ変換出力「いらっしゃい」

- ② 「ん」は、次の様にも入力できます。

「ん」 ..... 「**SHIFT** + **N**」または「**N**、**SHIFT** + **ろん**」

- ③ 「きゃ」、「ちえ」、「ぴゃ」などの文字は、大文字、小文字を別々に入力することもできます。

〔例〕「きゃ」 ..... 「**K**、**Y**、**A**」または「**K**、**I**、**SHIFT** + **Y A**」

↑  
大文字「き」入力

↑  
小文字「ゃ」入力

〔注〕上記の説明の中で、入力キーをカンマ（,）で区切った場合は、それらのキーを連続して入力することを示します。逆にプラス（+）で接続した場合は、それらのキーを同時に入力することを示します。

例「**N**、**SHIFT** + **ろん**」 ..... **N** キーを入力した後連続して、**SHIFT** キーを押しながら **ろん** キーを入力することを示します。

#### ※特殊文字の入力

入力方式がローマ字-カナ変換方式（**F5**）または**カナ**キーロックによるカナ入力方式の場合、日本語文書作成の上で欠かすことのできない特殊文字を入力することができます。

- ① 濁点「**ㇿ**」及び半濁点「**ㇾ**」の入力には、次のキーを押します。

「**ㇿ**」 ..... **ㇿ**

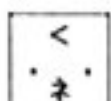
「**ㇾ**」 ..... **ㇾ**

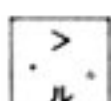
また間接出力モード（**F2**）の場合、変換フィールド内でひらがなやカタカナの直後に濁点又は半濁点を入力すると、自動的に濁点、半濁点のついた1文字のカナに変換されます。

〔例〕「し」の後に「ゐ」を入力すると、自動的に「じ」に変換されます。

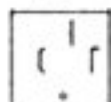
逆に直接モード（**F2**）を選択した場合、カナと濁点は各1文字分のスペースをとりますので、2文字として扱われます。

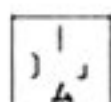
- ② 読点「、」及び句点「。」の入力には、次のキーを押します。

「、」……………**SHIFT** + 

「。」……………**SHIFT** + 

- ③ 始めかぎ括弧「〔」及び終わるかぎ括弧「〕」の入力には、次のキーを押します。

「〔」……………**SHIFT** + 

「〕」……………**SHIFT** + 

## 2 ミニ・ワープロ機能


本機のBASIC（CZ-8FB02）では、漢字変換機能だけでなく簡単なワードプロセッサとしての機能を持っています。この機能を利用すれば、日本語文章を画面に表示させたり、プリンタに印字させたりすることが、簡単に行なえます。本機のBASICでは、そのために、次のコマンド及びステートメントを用意しています。

LIST*	……………	画面表示用コマンド
LLIST*	……………	プリンタ印字用コマンド
CONSOLE#	……………	印字エリア設定用ステートメント
AUTO*	……………	REMステートメント付き行番号発生用コマンド

### (1) LIST\*

まず、アポストロフィ「'」（REMステートメントの省略形）を使って次のプログラムを入力してください。

```
10 ' パソコンテレビX1turbo
20 ' パーソナルコンピュータ
30 ' CZ-856C 新発売
```

ここでLIST\*を実行すると、次の画面のように、行番号や「'」もないふつうの文章になっています。

```
パソコンテレビX1turbo
パーソナルコンピュータ
CZ-856C 新発売
```



また、LIST\*の終わりに行番号を指定することによって、任意の行番号の文章だけの表示も可能です。

## (2)LLIST\*

LLIST\*コマンドは、アポストロフィ「'」に書かれている文章だけをプリンタに印字させる働きをします。したがってメッセージや説明文、手紙などの文章をアポストロフィ「'」を使って、プログラム中に作成することが可能となります。

## (3)CONSOLE#

CONSOLE#コマンドを使えばプリンタの印字エリアを設定することができますので、ハガキ大のものから大判サイズのものまで、自由に印字することができます。

## (4)AUTO\*

アポストロフィ「'」を使ったプログラムの作成には、AUTO\*コマンドを利用すると便利です。AUTO\*コマンドは、自動的にアポストロフィ「'」の行番号を発生させますので、プログラムということを意識することなく、文章を作成することができます。ただし、1つの行番号に入力できる文字数には限りがありますので注意してください（半角文字の場合250文字程度、全角文字の場合125文字程度）。

以上のように、本機のBASICでは多彩な日本語処理機能が追加されていますので、有効に活用すれば簡単なミニ・ワープロとして利用できます。

# 3 文字のコピー機能：COPY キー

## (1)画面上のコピー


本機の漢字BASICには、画面上に表示されている文字列（全角文字、半角文字を問わず）を任意の位置にそっくりコピーする機能があります。

コピーの方法を以下に示します。

①まず、コピーしたい位置にカーソルを移動し

**COPY** キーを押します。すると、カーソルの点滅が止まります。

②次に、カーソルコントロールキーでコピーしたい文章の先頭へカーソルを移動します。この時、移動したカーソルは点滅しています。


③キーを押すごとに、目的の位置に1文字ずつコピーされます。

途中でカーソルコントロールキーを押せば点滅している方のカーソルが移動しますので、任意の箇所からコピーを再開することができます。


④コピー先（点滅していない方のカーソル）が行割れをおこす（次の行になること）と自動


 けまして おめでとう。


↑ ②コピーしたい文章の先頭に  
↑ カーソルをもってくる。

 ←コピーしたい位置

① **COPY** キーを押す

あけまして  おめでとう。

あけまして 

③  キーを押すごとに  
コピーされる。

的にコピー機能が停止します。

また、コピーを中止したいときは再び **COPY** キーを押すか、**ESC** キーを押してください。

## (2)ハードコピー

**COPY** キーは、前述した画面上のコピー機能だけでなく、プリンタへのハードコピー機能も持っています。ハードコピーできる画面は、テキスト画面のみ、グラフィック画面のみ、テキスト画面とグラフィック画面を合成した画面の3通りです。ハードコピーを行なうためには、次のキーを入力します。

**SHIFT** + **COPY** .....テキスト画面のみハードコピー

**GRAPH** + **COPY** .....グラフィック画面のみハードコピー

**CTRL** + **COPY** .....テキスト画面とグラフィック画面を合成した画面のハードコピー

ただし以下の画面モードではハードコピーできません。

40文字×20行、80文字×20行、40文字×10行、80文字×10行





# 第2部

## 応用編

# プログラムの編集について

BASICプログラムを作成するときに、誤って隣のキーを押してしまったり、途中の文字を抜かしてしまったりする場合があります。また、行やステートメントを新しくつけ加えたり、今まであったものにとってしまったりする編集作業をすることがあります。このような時のために、いろいろな画面編集（スクリーンエディタ）機能が用意されています。

## 1 基本的な編集機能

### (1)カーソルの移動



●文字を修正したい時には、その修正箇所までカーソルを移動させます。カーソルを移動させるには以下のカーソルコントロールキーを使用します。







→を押すとカーソルは右へ移動します。画面の右端にカーソルがあるときにこのキーを押すと1つ下の行の左端に移動します。また、画面の一番下の行の右端にカーソルがあるときにこのキーを押すと、画面は1行上へスクロール（巻きあがり）し、カーソルは一番下の行の左端に移動します。

←を押すとカーソルは左へ移動します。画面の左端にカーソルがあるときにこのキーを押すとカーソルは1つ上の右端に移動します。また、画面の一番上の行の左端にカーソルがあるときにこのキーを押しても、カーソルは画面右はしに移動します。


↑を押すとカーソルは上へ移動します。画面の一番上の行にカーソルがあるときにこのキーを押してもカーソルは移動しません。

↓を押すとカーソルは下へ移動します。画面の一番下の行にカーソルがあるときにこのキーを押すと画面は1行上へスクロール（巻きあがり）します。

●通常キーボードのキーを押し続けると同じ文字あるいは同じ動作が継続して入力されます（リピート機能）。カーソルコントロールキーも押し続けると押している間カーソルが連続して移動します。REPEAT OFF と入力することにより、このリピート機能はなくなり、1文字ずつしか受け付けなくなります。再度リピート機能をはたらかせるにはREPEAT ON と入力します。

●カーソルを瞬時にホーム位置（画面の左上隅）に移動させるには  キーを押します。また  +  （  キーを押しながら  キーを押す）を押すとテキスト画面がすべて消され、カーソルはホーム位置に移動します。ただし、記憶されたプログラムは消えていませんのでLIST  とすれば画面に表示できます。

### (2)文字の削除

文字の削除を行なう場合には、削除したい文字の次の文字にカーソルを移動させた後、 キーを押します。この場合、カーソルの左にある文字が削除されカーソルから右の文字列が左へ移動します。



### (3)文字の挿入

文字の挿入を行なう場合には、文字を挿入したい位置の次の文字にカーソルを移動させた後 **SHIFT** + **INS DEL** キーを押します。この場合、カーソルから右の文字列が右へ移動し、カーソル位置にはスペース（空白）ができます。このスペース位置に追加したい文字を書き込みます。

このようなスクリーンエディタ機能を用いてプログラムの訂正を行なった場合には、各行番号の文の訂正が終わる度に、必ずキャリッジリターンキー (**↵**) を押してください。これにより、画面に表示されているステートメントの文字列がプログラムとしてメインメモリに登録されます。

(例) 「 10 PLIINT CH\$(65) 」を「 10 PRINT CHR\$(65) 」に訂正する場合次のようにしてください。

■はカーソルを示します。

10 PLIINT CH\$(65) ■

**←**キーを押してカーソルを「L」の位置に移動させ **R** キーを押します。

10 PR**I**INT CH\$(65)

**←**キーを押してカーソルを次の「I」の位置に移動させ **INS DEL** キーを押して「I」を1文字削除します。

10 PR**I**NT CH\$(65)

**←**キーを押してカーソルを「\$」の位置に移動させ **SHIFT** + **INS DEL** キーを押して「H」と「\$」の間に1文字分のスペースを作ります。

10 PRINT CH■\$(65)

**R**キーを押して「H」と「\$」の間に文字「R」を挿入します。

10 PRINT CHR**\$**(65)

**↵**キーを押してプログラムをメインメモリへ登録します。

以上のスクリーンエディタ機能を利用してプログラムの修正は行なえますが、画面編集をより効果よく迅速に行なうために、さらに便利な機能として次のような機能が用意されています。

## 2 知っていると便利な編集機能

### (1)カーソルの移動

#### ●水平TAB

画面の初期状態では水平TAB位置は画面左端から8文字単位に設定されています。**H TAB** キーを押すとカーソルは現在位置から次の水平TAB位置（初期状態では8文字右）へ瞬時に移動します。この機能を利用すると、文字の始まり位置を統一することが容易に行なえます。また、水平TAB位置は表示画面内で任意に設定できます。まず、設定されているTAB位置を取り消す場合にはその取り消す水平TAB位置へカーソルを移動させ **CTRL** + **Y** キーを押します。逆に新しく水平TABを設定する場合には、設定したい水平位置にカーソルを移動させ **CTRL** + **T** キーを押します。

#### ●1ワード単位でのカーソル移動

**CTRL** + **F** キーを押すとカーソルは現在位置より右（先）にあるワードの先頭に移動します。また、**CTRL** + **B** キーを押すとカーソルは現在位置より左（後）にあるワードの先頭に移動します。ここで、ワードとは空白、記号を除く連続した文字列のことです。ただし、コロン (:) はワードとみなされますが、πはワードとはみなされません。



## (2)文字の削除

### ●カーソル位置からその行の終わりまでの削除

**CTRL** + **E** キーを押すとカーソル位置からその行の終わりまでの文字が削除されます。この後で、キャリッジリターンキーを押すことによりプログラムが修正されます。

### ●カーソルのある位置以降の画面クリア

**CTRL** + **Z** キーを押すとカーソル位置以降の画面をすべてクリアします。

## (3)文字の挿入

文の途中で何文字か挿入したい場合には、文字を挿入したい位置の次の文字にカーソルを移動させ **CTRL** + **A** キーを押します。これで、インサートモードに設定され、それ以後に入力された文字がカーソル位置に表示されると共にカーソルより右の文字列が右へ移動していきます。インサートモードを解除するには **CTRL** + **S** キー (BREAKと同じ) を押します。また、キャリッジリターンキーを押してもインサートモードは解除されます。また、インサートモード時に **INS DEL** キーを押して文字を削除してもインサートモードは解除されません。

## (4)行の分割

1行の内容を2行にわけるように書き直したい場合には、区切りたい箇所へカーソルを移動させ **CTRL** + **J** キーを押すと、カーソルからその行の終わりまでの文が、自動的に次の行へ移動します。移動した行の先頭に行番号をつけてキャリッジリターンキーを押せばこれまでの1行の内容が2行にわたって書かれます。

(例) 次に示す行番号50で定義されているプログラムをCIRCLE文とPAINT文に分離してみましょう。

```
50 CIRCLE(100,100),50,1:PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```


カーソルをPAINT文の「P」の位置に移動させ **CTRL** + **J** キーを押します。するとPAINT文以降が次の行へ移動します。

```
50 CIRCLE(100,100),50,1:
```

```
PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```

**SHIFT** + **INS DEL** キーを押してPAINT文の前に行番号を挿入するためのスペースを作り、行番号60を入力します。

```
60 PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```

キャリッジリターンキー  を押して行番号50, 60の文をプログラムとしてメインメモリへ登録します。

## (5)行の結合

2行にわたる文を1行に納めたい場合には、行の結合機能を利用します。**CTRL** + **W** キーを押すと、カーソルのある水平1行と次の1行とが結合されます。よって2行にわたる文が1行として扱われ、これまでの2行の文と文の間隔は **INS DEL** キーを押して文字を削除することによりつめることができます。結合する2つの行のうち前の1行の文が画面の数行にわたるときには、カーソルをその文の1行内の最下行に移動させて **CTRL** + **W** キーを押してください。

(例) 次に示す行番号50と60で定義されているプログラムを1つの行番号の文にまとめてみましょう。



```
50 CIRCLE(100,100),50,1:  
60 PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```

行番号50のライン上にカーソルを移動させ **CTRL** + **W** キーを押します。これにより行番号50と60の文が結合されました。この時点ではまだ、メインメモリへは登録されていません。

```
50 CIRCLE(100,100),50,1:  
60 PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```

カーソルをPAINT文の「P」の位置に移動させ **INS DEL** キーを押すとPAINT文が左へ移動し画面左端まで移動します。さらに押し続けると1行上の行番号50行の画面右端よりPAINT文が現われます。CIRCLE文の右にPAINT文を並べた時点で **ENTER** キーを押します。

```
50 CIRCLE(100,100),50,1:PAINT(100,100),HEXCHR$("AA00AA00AA00"),1
```

これにより、文番号50にCIRCLE文とPAINT文がプログラムとしてメインメモリへ登録されました。しかし文番号60にはまだPAINT文が残った状態となっていますので文番号60の文を消す必要があります。

```
50 CIRCLE(100,100),50,1:PAINT(100,100),HEXCHR$("AA00AA00AA00"),1  
60 ENTER
```

文番号60のみを入力してキャリッジリターンキーを押すことにより文番号60のプログラムがメインメモリから削除されます。以上により2つの行番号で定義されていたプログラムが1つの文番号内に定義されました。

## (6)文字のコピー

本機は画面に表示されている文字をそのまま別の場所にコピーして表示する機能を持っています。コピーをするにはまず、コピーする文字を表示したい場所にカーソルを移動し、**COPY** キーを押します。カーソルは点滅を停止し静止した状態になります。ここで、カーソルコントロールキーを押すと、静止したカーソル位置から点滅したカーソルが現われます。このカーソルをコピーしたい文字の先頭まで移動させます。**ENTER** キーを押すと、点滅したカーソル位置にある文字が点滅を停止したカーソル位置へコピーされ、両カーソルは右へ移動します。コピーを終了するには再度 **COPY** キーを押します。

このコピー機能によりコピー表示する行は水平1行にのみ有効です。次の行にわたる必要がある場合には以上のコピー操作を再度行なってください。

また、この **COPY** キーは次のようにプリンタへの画面コピー機能を持っています。

**SHIFT** + **COPY** キーを押すとテキスト画面のみをプリンタにコピーします。

**GRAPH** + **COPY** キーを押すとグラフィック画面のみをプリンタにコピーします。

**CTRL** + **COPY** キーを押すとテキスト画面とグラフィック画面をプリンタにコピーします。

ただし以下の画面モードの時には画面コピーはできません。



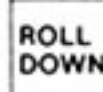
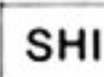



40文字×20行、80文字×20行、40文字×10行、80文字×10行



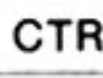



## (7)行間への新しい行の追加

表示画面の行と行の間をひろげ新しく行を追加したい場合には、**ROLL UP** キーあるいは **ROLL DOWN** キーを使用します。**ROLL UP** キーを押すとカーソルのある行から上の行が1行上にスクロールします。また、**ROLL DOWN** キーを押すとカーソルのある行から下の行が1行下にスクロールします。これにより、行と行の間がひろげられ新しい行を追加できます。ただし、次の項(8)に示すように、エディットモードではこれらのキーの働きが異なります。



## (8)EDIT機能

プログラムを実行させた場合、プログラムに何らかのエラーがあると、エラーの種類とそのエラーが生じた行番号を表示して実行が中断されます。このような場合EDIT文を使用することによりエラーの発生したプログラムの修正を能率よく行えます。例えばEDIT  と入力すると、エラーの発生した行を画面に表示して、カーソルはその先頭に表示され、エディットモードに入ります。エディットモードでは  キーあるいは  キーを使用することにより、巻き物を見るように、画面に表示されている文の前後を表示できます。この機能を利用して、訂正する文を画面に表示させ訂正します。エディットモードを解除するには  +  キー あるいは  +  キーを押してください。

エディットモードで  キーまたは  キーを使用して、プログラムを画面全体に表示してスクロールしている状態で、新しく行を追加するには  +  あるいは  +  キーを使用して画面に空白部分を作成してから行番号付文をキー入力し、最後にキャリッジリターンキーを入力してください。

また、EDIT機能はエラーが発生した時のみでなく、プログラムの作成後、プログラムの見直しを行う場合に便利です。LIST文の実行でもプログラムの見直しはできますが、画面上部へ消えていった行を表示するには再度LIST文を実行する必要があります。

EDIT文は次の形式で表わします。







EDIT   n   .  

n…行番号

.…ピリオド(.)にはエラーが発生したときの行番号が入ります。

## (9)行番号の整理

プログラムを修正して行を取り除いたり、新しい行を追加すると行番号がばらばらになります。このようなとき、RENUM文を使用して行番号を整理することができます。RENUM文は次の形式で表わし、旧行番号mで指定した行以降の行番号を新行番号lで始まる行番号に増分nで付け変えます。この場合、プログラム内のGOTO文などでジャンプする行番号も新しい行番号に書き変わります。

RENUM  l  ,  m  ,  n 

l…新行番号。省略すると10。

m…旧行番号。省略するとプログラムの最初の行。

n…増分。省略すると10。

# 3 全角文字の編集

本機は一般の英数カナ文字に加えて、日本語文字も簡単にテキスト画面に表示することができ、プログラム中でも日本語文字を使用できます。ただし、標準解像度モード(縦200ライン表示モニター用)でテキスト画面が40×25、40×20、80×25、80×20行モード時には日







本語文字は表示できません。一般の英数カナ文字1バイトコードで表わされて**半角文字**と呼ばれるのに対して、日本語文字は2バイトコードで表わされ**全角文字**と呼ばれ、文字の大きさも英数カナ文字の水平方向が2倍の大きさで表示されます。

プログラムを編集する際に、全角文字は半角文字と取り扱い方やカーソルの動作が異なります。以下に、全角文字の編集方法について説明します。


### (1)全角文字の表示

プログラムの作成中に全角文字を使用する場合には、まず **CTRL** + **XFER** キーを押して日本語入力モードにします。画面の最下行に変換フィールドが表われますので、表示したい全角文字を指定してキャリッジリターンキーを押すとカーソルの位置に指定した全角文字が2桁（1桁は半角文字1文字分を示します）の大きさで表示されます。再度 **CTRL** + **XFER** キーを押すと日本語入力モードが解除されます。日本語入力モードに関しては「5章. 日本語処理」の項を参照してください。


### (2)全角文字上のカーソルの移動

カーソルコントロールキー  を操作してカーソルを全角文字上に移動するとカーソルは全角文字全体を覆う大きさに変わります。これは全角文字の1桁目にカーソル位置があることを示します。もう1回  キーを押すとその全角文字の右半分のみの大きさに変わります。これは全角文字の2桁目にカーソル位置があることを示します。このように、全角文字列の中では  キーを2回押すことによりカーソルは1文字進みます。カーソルを  キーにより戻す場合にはカーソルはこの逆の動作を行ないます。

### (3)全角文字の削除


全角文字の削除を行なう場合には、削除したい全角文字の次の文字にカーソルを移動（次の文字が全角文字の場合にはカーソル位置はその文字の前半でも後半でもかまいません）させた後、 キーを押します。この場合、カーソルの左にある全角文字が削除されカーソルから右の文字列が左へ1文字分（2桁）移動します。

### (4)全角文字の挿入

文字列の途中に全角文字を挿入する場合には、挿入したい位置の次の文字にカーソルを移動（次の文字が全角文字の場合には、カーソル位置はその文字の前半でも後半でもかまいません）させた後、**SHIFT** +  キーを2回押します。この場合、カーソルから右の文字列が右へ2桁移動し、全角文字1文字分のスペースができます。このスペースに追加したい全角文字を書き込みます。

### (5)全角文字の修正

入れまちがえた全角文字を正しい全角文字に置き換えたい場合には、まちがった全角文字の前半の位置にカーソルを移動させた後、正しい全角文字を入力します。もし、カーソルが全角文字の後半にあった場合に新しく全角文字を入力すると、修正したい全角文字の次にある文字を消してしまいます。

このような機能を利用してプログラムの訂正を行った場合には各行番号の文の訂正が終わる度に、必ずキャリッジリターンキー () を押してください。これにより、画面に表示されている

ステートメントの文字列がプログラムとしてメインメモリに登録されます。

(例) 「10 PRINT"本日語の文を使います。"」を

「10 PRINT"日本語文字を使います。"」に訂正する場合次のようにしてください。

■はカーソル位置を示します。

10 PRINT"本日語の文を使います。" ■

☞ キーでカーソルを「本」の位置まで移動します。

10 PRINT"☞日語の文を使います。"

COPY キーを押した後、点滅するカーソルを「日」の位置に移動させて☞キーを押して「日」を「本」の位置にコピーした後、再度 COPY キーを押してコピーモードを解除します。

10 PRINT"日☞語の文を使います。"

SHIFT + XFER キーを押して日本語入力モードにした後、全角文字、ローマ字入力、音訓変換モードになっていることを確認して H、O、X、XFER と入力して「本」を指定して、☞キーを押します。

10 PRINT"日本☞の文を使います。"

☞ キーを4回押してカーソルを「文」の位置に移動させます。

10 PRINT"日本語の☞を使います。"

INS DEL キーを1回押して「の」を削除します。

10 PRINT"日本語☞を使います。"

☞ キーを2回押してカーソルを「を」の位置に移動させます。

10 PRINT"日本語文☞使います。"

SHIFT + INS DEL キーを2回押して「文」と「を」の間に全角文字1字分のスペースを作ります。

10 PRINT"日本語文 ☞ 使います。"

日本語入力モードの状態で J、I、XFER と入力して「字」を指定して☞キーを押します。SHIFT + XFER キーを押して日本語入力モードを解除します。

10 PRINT"日本語文字☞使います。"







☞ キーを押して修正したプログラムをメインメモリへ登録します。



## 4

## コントロールコード表

以下にコントロールコードの一覧表を示します。なお下の表で **CTRL** + A は **CTRL** キーを押しながら **A** キーを押すことを意味します。

CTRL+	コード	処 理 内 容
@	00	ダミー。
A または a	01	インサートモード <sup>1)</sup> にする。解除は <b>BREAK</b> または  で行なえる。
B または b	02	現在のワード <sup>2)</sup> の先頭にカーソルを戻す。
C または c	03	実行を停止する。( = <b>SHIFT</b> + <b>BREAK</b> )
D または d	04	画面などを標準の状態に戻す。( = <b>INIT</b> )
E または e	05	現在のカーソル以降1行を消す。
F または f	06	次のワードの先頭にカーソルを移す。
G または g	07	ビープ音を鳴らす。( = <b>BEEP</b> )
H または h	08	1文字分消して戻る。( = <b>INS DEL</b> )
I または i	09	水平タブレーションを行なう。( = <b>H TAB</b> )
J または j	0A	現在のカーソル以降を次の行に分ける。
K または k	0B	カーソルを画面のホーム位置に移す。( = <b>CLR HOME</b> )
L または l	0C	テキスト画面を消去する。( = <b>SHIFT</b> + <b>CLR HOME</b> )
M または m	0D	キャリッジターンをする。( =  )
N または n	0E	現在のカーソルから上を上方向にスクロールする。( = <b>ROLL UP</b> )
O または o	0F	現在のカーソルから下を下方向にスクロールする。( = <b>ROLL DOWN</b> )
P または P	10	ダミー。
Q または q	11	実行の一時停止を解除する。
R または r	12	空白を挿入する。( = <b>SHIFT</b> + <b>INS DEL</b> )
S または s	13	実行を一時停止する。 <sup>3)</sup> ( = <b>BREAK</b> )
T または t	14	水平タブレーション位置の新たな設定を行なう。
U または u	15	ダミー。
V または v	16	ダミー。
W または w	17	現在のカーソルがある行と次の行をつないで1つにする。
X または x	18	ダミー。
Y または y	19	現在のカーソル位置の水平タブレーションの解除を行なう。
Z または z	1A	現在のカーソルより下のテキスト画面をすべて消去する。
[	1B	ダミー。
¥	1C	カーソルを右へ移動する。( =  )
]	1D	カーソルを左へ移動する。( =  )
^	1E	カーソルを上へ移動する。( =  )
_	1F	カーソルを下へ移動する。( =  )
0		画面の背景色を黒(透明)にする。( = <b>COLOR, 0</b> )
1		画面の背景色を青にする。( = <b>COLOR, 1</b> )
2		画面の背景色を赤にする。( = <b>COLOR, 2</b> )
3		画面の背景色をマゼンタにする。( = <b>COLOR, 3</b> )
4		画面の背景色を緑にする。( = <b>COLOR, 4</b> )
5		画面の背景色をシアンにする。( = <b>COLOR, 5</b> )
6		画面の背景色を黄色にする。( = <b>COLOR, 6</b> )
7		画面の背景色を白にする。( = <b>COLOR, 7</b> )



テンキーの	0	文字グラフィックの色を黒(透明)にする。(=COLOR 0)
	1	文字グラフィックの色を青にする。(=COLOR 1)
	2	文字グラフィックの色を赤にする。(=COLOR 2)
	3	文字グラフィックの色をマゼンタにする。(=COLOR 3)
	4	文字グラフィックの色を緑にする。(=COLOR 4)
	5	文字グラフィックの色をシアンにする。(=COLOR 5)
	6	文字グラフィックの色を黄色にする。(=COLOR 6)
	7	文字グラフィックの色を白にする。(=COLOR 7)
	/	キャラクタジェネレータのROM/RAMを切り換える。(=CGEN)
	*	文字を点滅モードとノーマルモードとに切り換える。(=CFASH)
	-	文字を反転モードとノーマルモードとに切り換える。(=CREV)

- 1) インサートモード (insert mode) 挿入したい文字キーを押すたびに、カーソルから右の部分が自動的に右に移動して、キーボードから入力した文字が挿入されるモード。
- 2) ワード (word) 隙間のない英数字の文字列をいいます。
- 3) プログラムの実行中は、キーを押している間だけ停止し、キーを離すと再び実行を開始します。

## 2章

# ディスプレイモード

## 2章

BASICは、図形文字（アルファベット、数字、漢字、カタカナ、ひらがな、セミグラフィック文字など）、点、線、その他複雑な図形をディスプレイテレビに表示することができます。

アルファベットなどの図形文字を表示する画面を**テキスト画面**、ドットによる図形を表示する画面を**グラフィック画面**と呼び、ディスプレイテレビの画面は、この2種類の画面を重ねて表示していると考えることができます。

### 1

## テキスト画面

テキスト画面に表示できる文字数は、使用するディスプレイが標準ディスプレイか、高解像度ディスプレイかにより異なります。

○標準ディスプレイ（文字数／行）×（行数／画面）

40文字×25行、80文字×25行

40文字×12行、80文字×12行

40文字×20行、80文字×20行

40文字×10行、80文字×10行

○高解像度ディスプレイ

40文字×25行、80文字×25行

40文字×12行、80文字×12行

40文字×20行、80文字×20行

ディスクBASICを起動した直後は

標準ディスプレイ使用時には 80文字×12行


高解像度ディスプレイ使用時には 80文字×25行

に設定されています。

テキスト画面の表示文字数は、WIDTHステートメントを使います。

WIDTH [1行当たりの文字数], [画面に表示する行数]

たとえば、40文字×25行の表示画面に変更するには

WIDTH 40, 25 

と入力します。すると画面をクリアした後、40文字×25行の画面に変わります。WIDTHステートメントを使って、次の8通りの設定ができます。

ステートメント	表示文字数	備 考
WIDTH40, 10, g, d	40×10	標準ディスプレイモードのときのみ設定可能。 アンダーラインが引ける。 グラフィック画面表示不可。
WIDTH80, 10, g, d	80×10	
WIDTH40, 12, g, d	40×12	
WIDTH80, 12, g, d	80×12	
WIDTH40, 20, g, d	40×20	アンダーラインが引ける。 グラフィック画面表示不可。
WIDTH80, 20, g, d	80×20	
WIDTH40, 25, g, d	40×25	
WIDTH80, 25, g, d	80×25	

※ g：グラフィック画面の解像度の設定

0 = 200 / 192ドット 1 = 400 / 384ドット

d：ディスプレイモードの設定

0 = 本体の標準／高解像度切換スイッチの状態に従う。

■：標準ディスプレイモード (STANDARD)

■：高解像度ディスプレイモード (HIGH)

1 = 標準ディスプレイモード

2 = 高解像度ディスプレイモード

(注) ・標準ディスプレイでは、標準ディスプレイモードのみ、高解像度ディスプレイでは、高解像度ディスプレイモードのみ使用可能です。

・専用ディスプレイテレビ(CZ-855DまたはCZ-850D)では、標準ディスプレイモードおよび高解像度ディスプレイモードのどちらでも使用可能です。

その切換えは、WIDTHステートメントの第4パラメータdによって行ないます。

## 2 グラフィック画面

### (1) 横方向の解像度

WIDTHステートメントの第1パラメータによって決まります。

WIDTH 40 ならば 320ドット

WIDTH 80 ならば 640ドット

### (2) たて方向の解像度

本機では、グラフィック画面は、2つのグラフィックメモリ(48KBが2つ)から構成されています。本機の専用ディスプレイテレビ(CZ-855DまたはCZ-850D)は

標準ディスプレイモード (たて200ドット)

高解像度ディスプレイモード (たて400ドット)

の切換えができます。

これは、WIDTHステートメントの第3、4パラメータで指定します。

指定の方法は次のとおりです。



たてのドット数	第3パラメータの値	第4パラメータの値
200	0	1
400	1	2

ただし、たて400ドットを指定する場合は、あらかじめ

**OPTION SCREEN 0**

を実行してください。

※本機のグラフィックメモリは、本来のグラフィック表示以外に、データやプログラムを記憶する普通のメモリとしても使用できます。これを指定するのがOPTION SCREENステートメントです。詳しくは「6章 フリーエリアについて」を参照してください。

OPTION SCREEN 0…全部グラフィック用に使用。たて400ライン可


OPTION SCREEN 1…半分を変数エリア用に転用。たて200ライン可 (BASIC起動時)


OPTION SCREEN 2…変数エリアと外部記憶用に使用。

OPTION SCREEN 3…半分を外部記憶用に転用。たて200ライン可

OPTION SCREEN 4…全部外部記憶用に使用。

なお、第4パラメータを0にして、本機前面ドア内の標準／高解像度切換スイッチによって


標準ディスプレイモード (STANDARD) 


高解像度ディスプレイモード (HIGH) 

に切りかえることができます。

以上を次表にまとめます。

ステートメント	解 像 度	使用ページ数
WIDTH40, 25, 0, d	320×200	カラー4 / 白黒12
WIDTH80, 25, 0, d	640×200	カラー2 / 白黒6
WIDTH40, 12, 0, d	320×192	カラー4 / 白黒12
WIDTH80, 12, 0, d	640×192	カラー2 / 白黒6
WIDTH40, 25, 0, D	320×200	カラー4 / 白黒12
WIDTH80, 25, 0, D	640×200	カラー2 / 白黒6
WIDTH40, 12, 0, D	320×192	カラー4 / 白黒12
WIDTH80, 12, 0, D	640×192	カラー2 / 白黒6
WIDTH40, 25, 1, D	320×400	カラー2 / 白黒6
WIDTH80, 25, 1, D	640×400	カラー1 / 白黒3
WIDTH40, 12, 1, D	320×384	カラー2 / 白黒6
WIDTH80, 12, 1, D	640×384	カラー1 / 白黒3

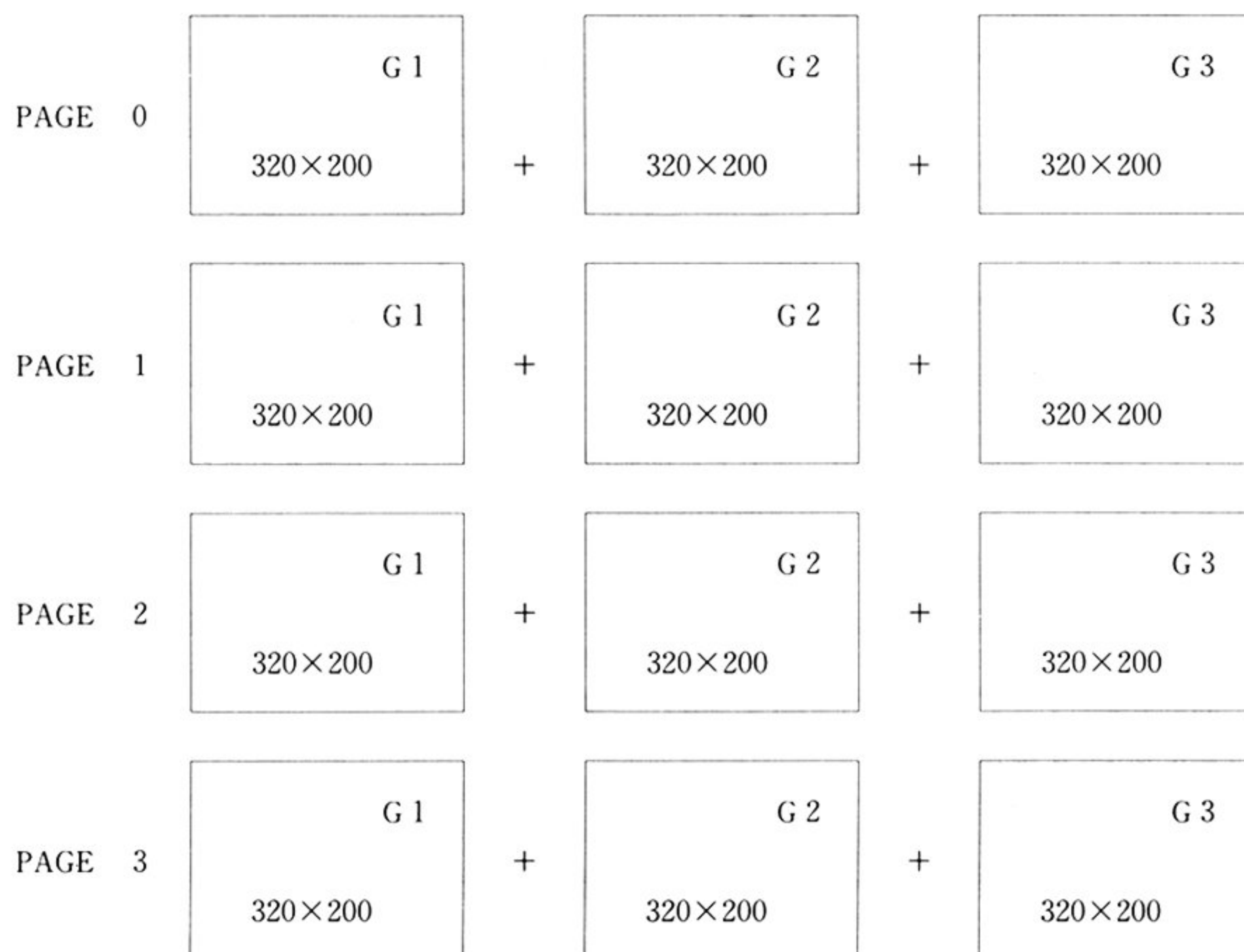
d: 0または1 (標準ディスプレイモードに設定。d=0のとき、本体の標準／高解像度切換スイッチは標準ディスプレイモード (  ) にセットされていること)

D: 0または2 (高解像度ディスプレイモードに設定。D=0のとき、本体の標準／高度解像度切換えスイッチは高解像度ディスプレイモード (  ) にセットされていること)

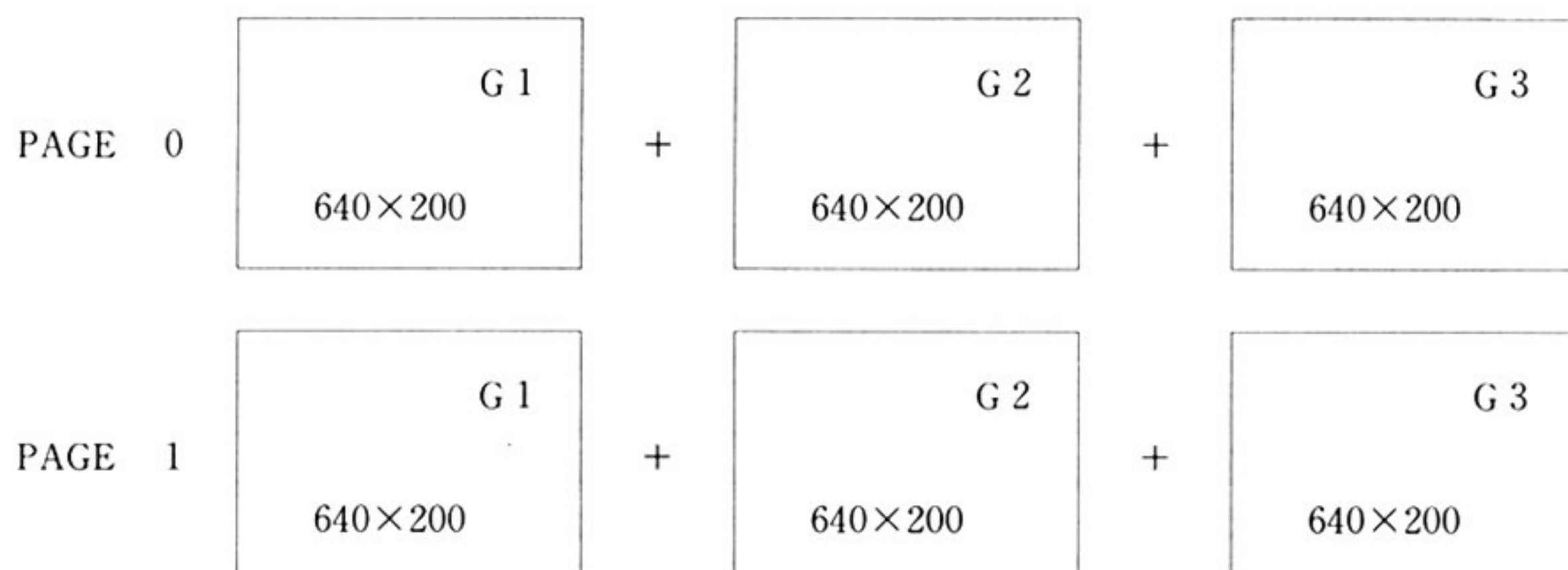
グラフィック画面の基本的な構成図を次に示します。

本書では、1枚目のグラフィック画面をG 1、2枚目の画面をG 2、3枚目の画面をG 3と呼んで区別しています。

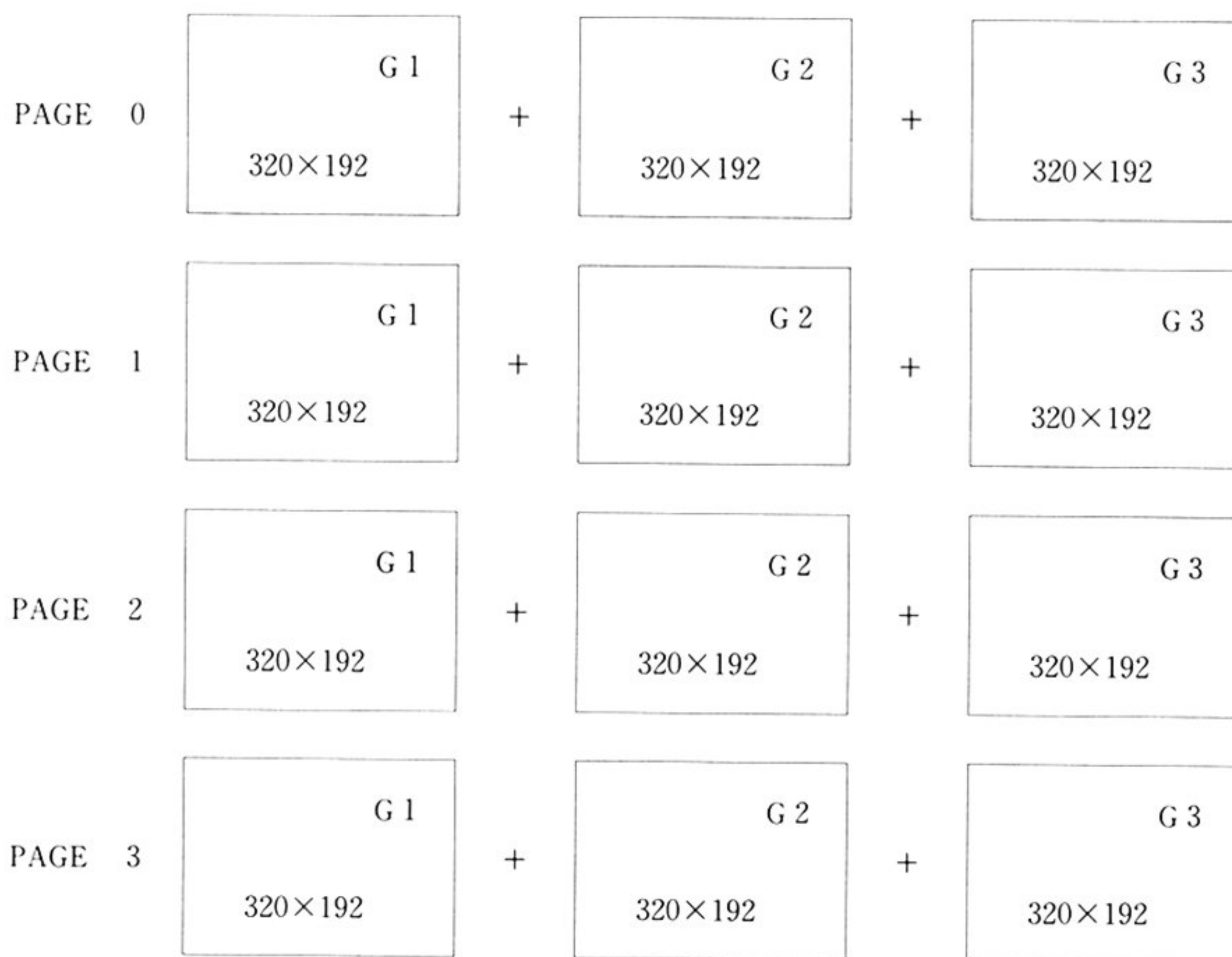
〈高解像度／標準ディスプレイモードで、WIDTH 40, 25, 0, Dを設定 (D=0または1または2)〉



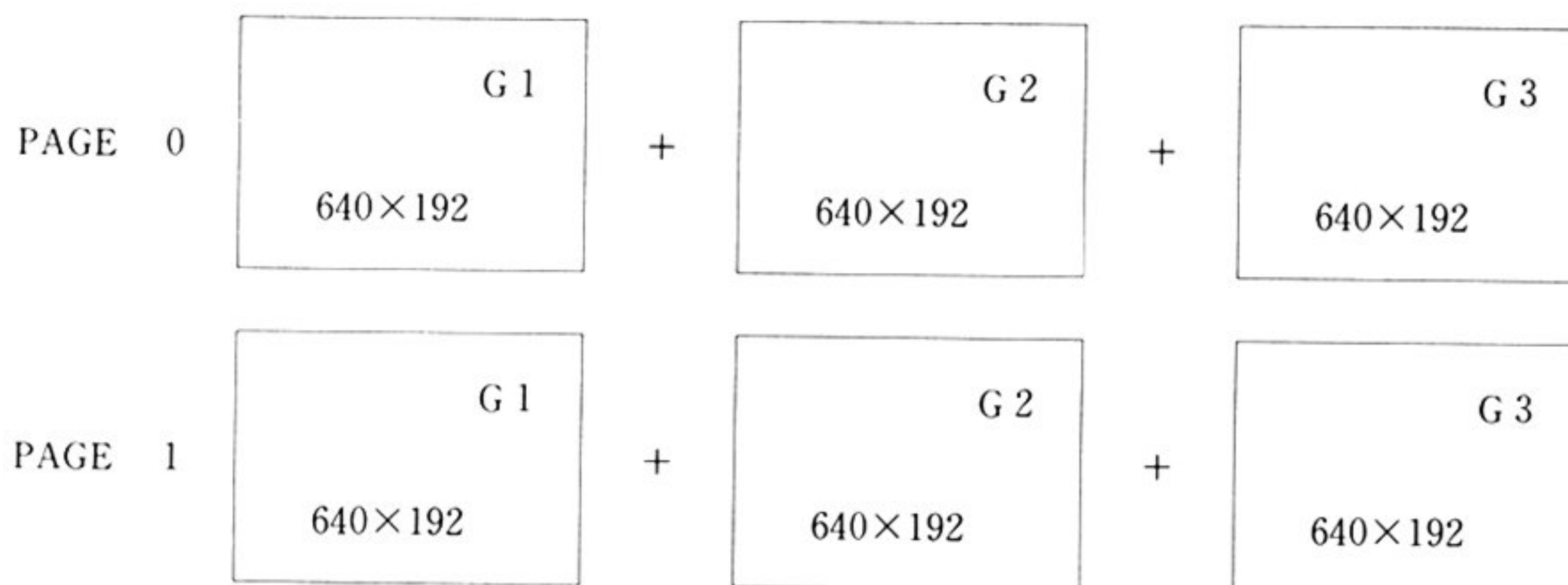
〈高解像度／標準ディスプレイモードで、WIDTH 80, 25, 0, Dを設定 (D=0または1または2)〉



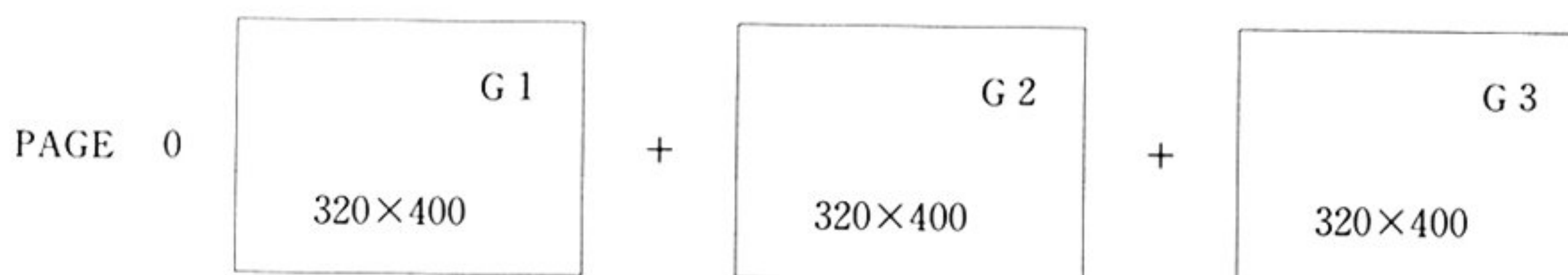
〈高解像度／標準ディスプレイモードで、WIDTH40,12,0,Dを設定 (D=0または1または2)〉



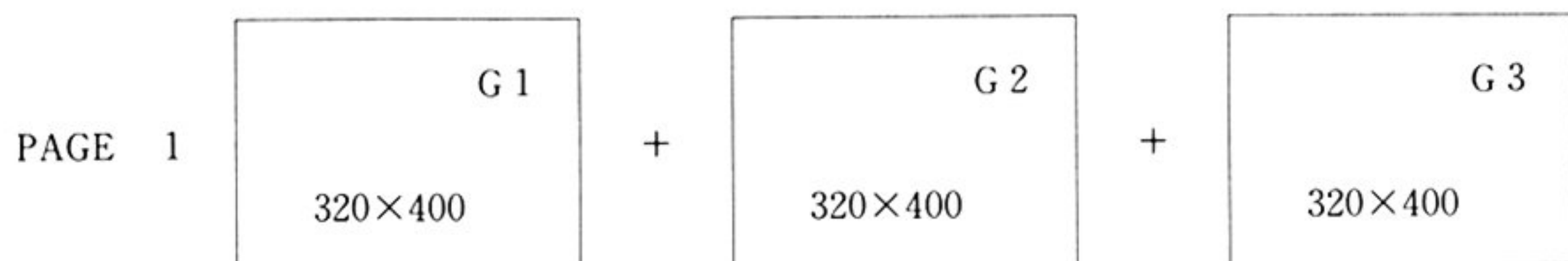
〈高解像度／標準ディスプレイモードで、WIDTH80,12,0,Dを設定 (D=0または1または2)〉



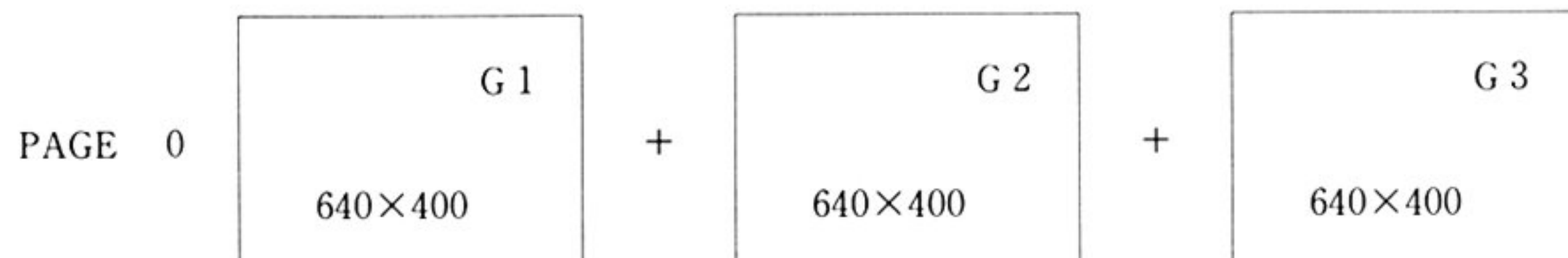
〈高解像度ディスプレイモードで、WIDTH40,25,1,Dを設定 (D=0または2)〉



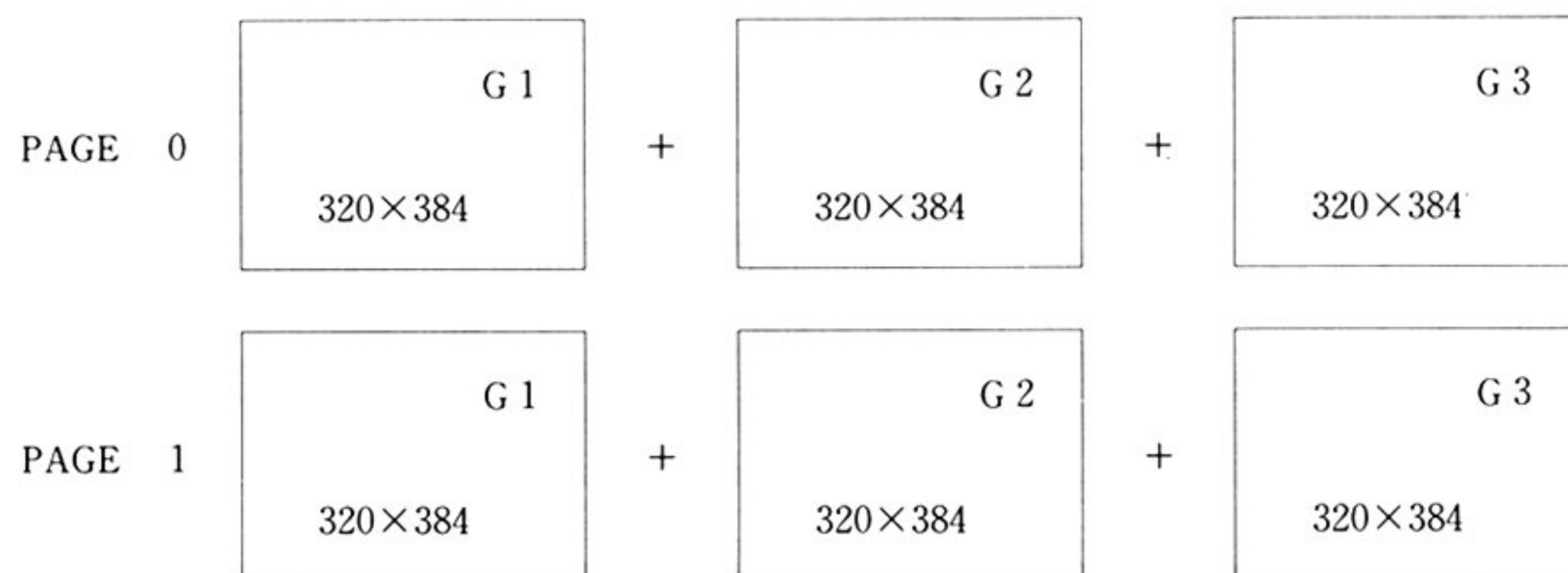




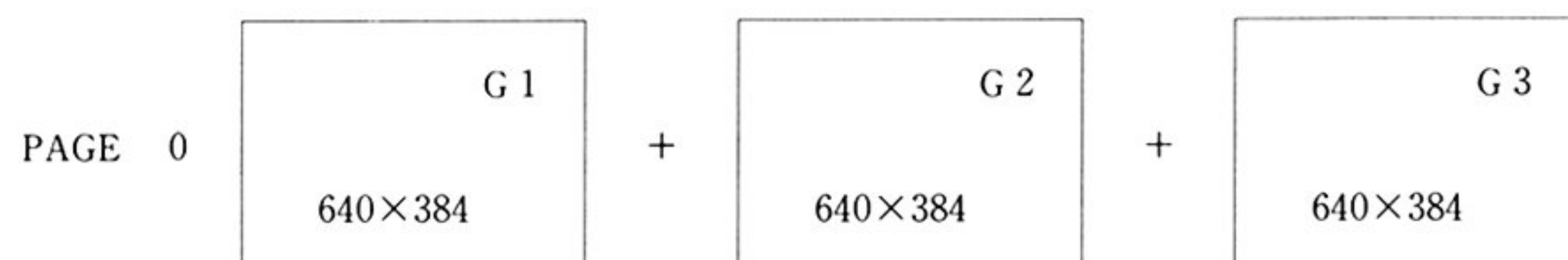
〈高解像度ディスプレイモードで、WIDTH 80, 25, 1, Dを設定 (D=0または2)〉



〈高解像度ディスプレイモードで、WIDTH 40, 12, 1, Dを設定 (D=0または2)〉



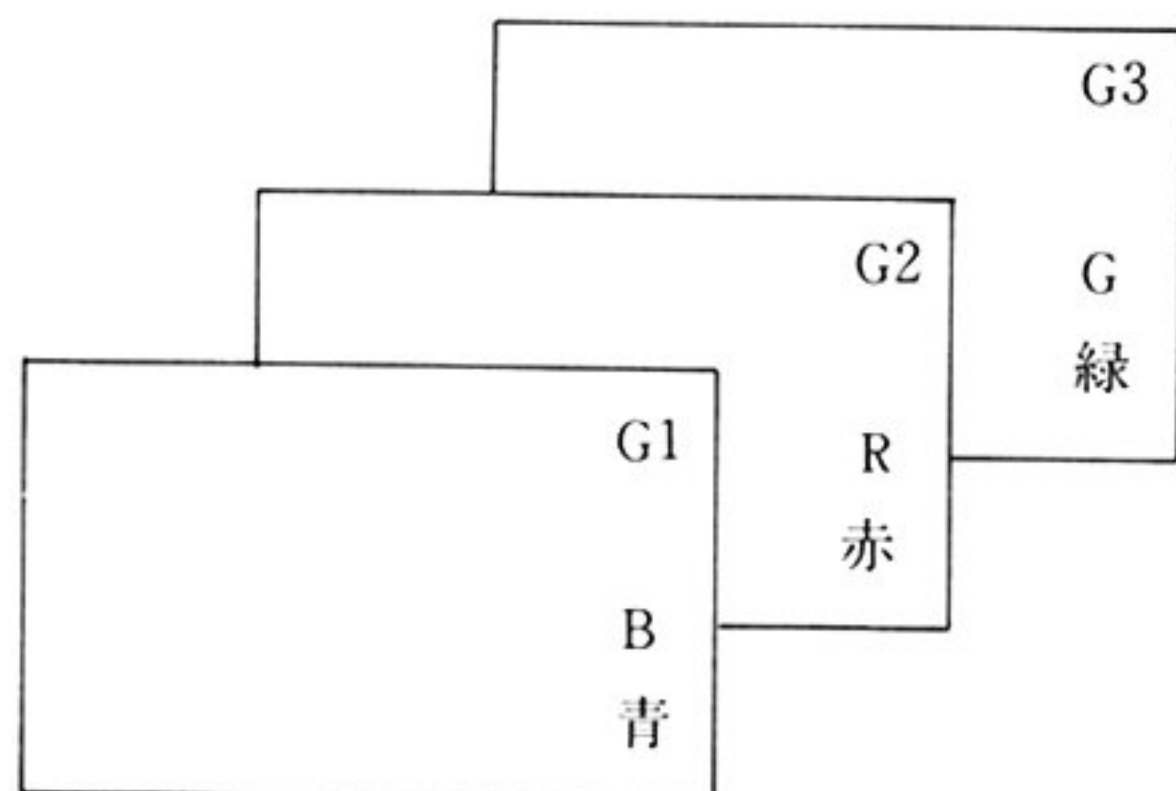
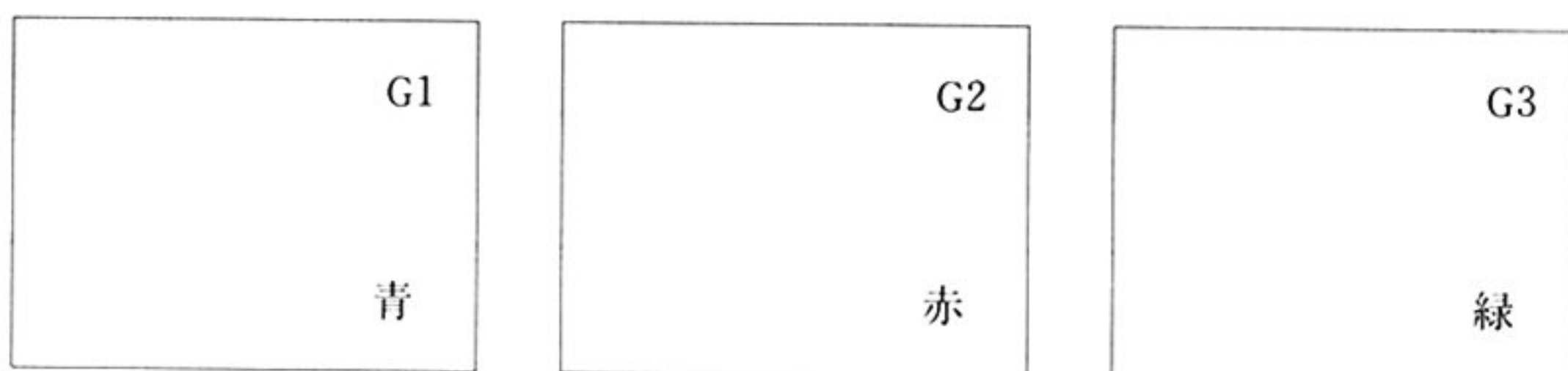
〈高解像度ディスプレイモードで、WIDTH 80, 12, 1, Dを設定 (D=0または2)〉



※ 画面内の積は、(横のドット数) × (たてのドット数) を表わす。

G 1、G 2、G 3の各グラフィック画面は、次の図のようにG 1が青、G 2が赤、G 3が緑というように設定されています。この3枚のグラフィック画面は合成されて画面上に表示されます。たとえば、青の画面にドットがあり、赤と緑の画面にドットがない場合はそのドットは青で表示されます。また、青と赤の画面にドットがあり、緑の画面にドットがない場合は、そのドットはマゼンタ（紫）で表示されます。

(カラーテレビの画面は、青、赤、緑の3原色を合成したものですが、BASICのカラーグラフィックも、同様に、青、赤、緑の3枚の画面を重ねて表示したものです。)



3枚のグラフィック画面の合成  
(B R G 合成)

### 3

## カラーコード

カラーコードは0～7の整数で表わされ、その値によって画面の色が設定されます。

次の図はG 1、G 2、G 3の画面とカラーコードの関係を示しています。

このようにカラーコードとは、青の画面をビット0、赤の画面をビット1、緑の画面をビット2とした3ビットの2進数を10進数表現したものとなっています。

色	G3	G2	G1	2進数ビット表現	カラーコード
黒 (透明)	○	○	○	0 0 0	0
青	○	○	●	0 0 1	1
赤	○	●	○	0 1 0	2
マゼンタ	○	●	●	0 1 1	3
緑	●	○	○	1 0 0	4
シアン	●	○	●	1 0 1	5
黄	●	●	○	1 1 0	6
白	●	●	●	1 1 1	7

※ ●はドットがある、○はドットがないことを示しています。

## 4

## パレットコード

カラーコードは色そのものを指定するコードで、

0 = 黒 (透明)、1 = 青、2 = 赤、3 = マゼンタ、4 = 緑、5 = シアン、6 = 黄、7 = 白  
 というように、数字と色が絶対的に対応しています。

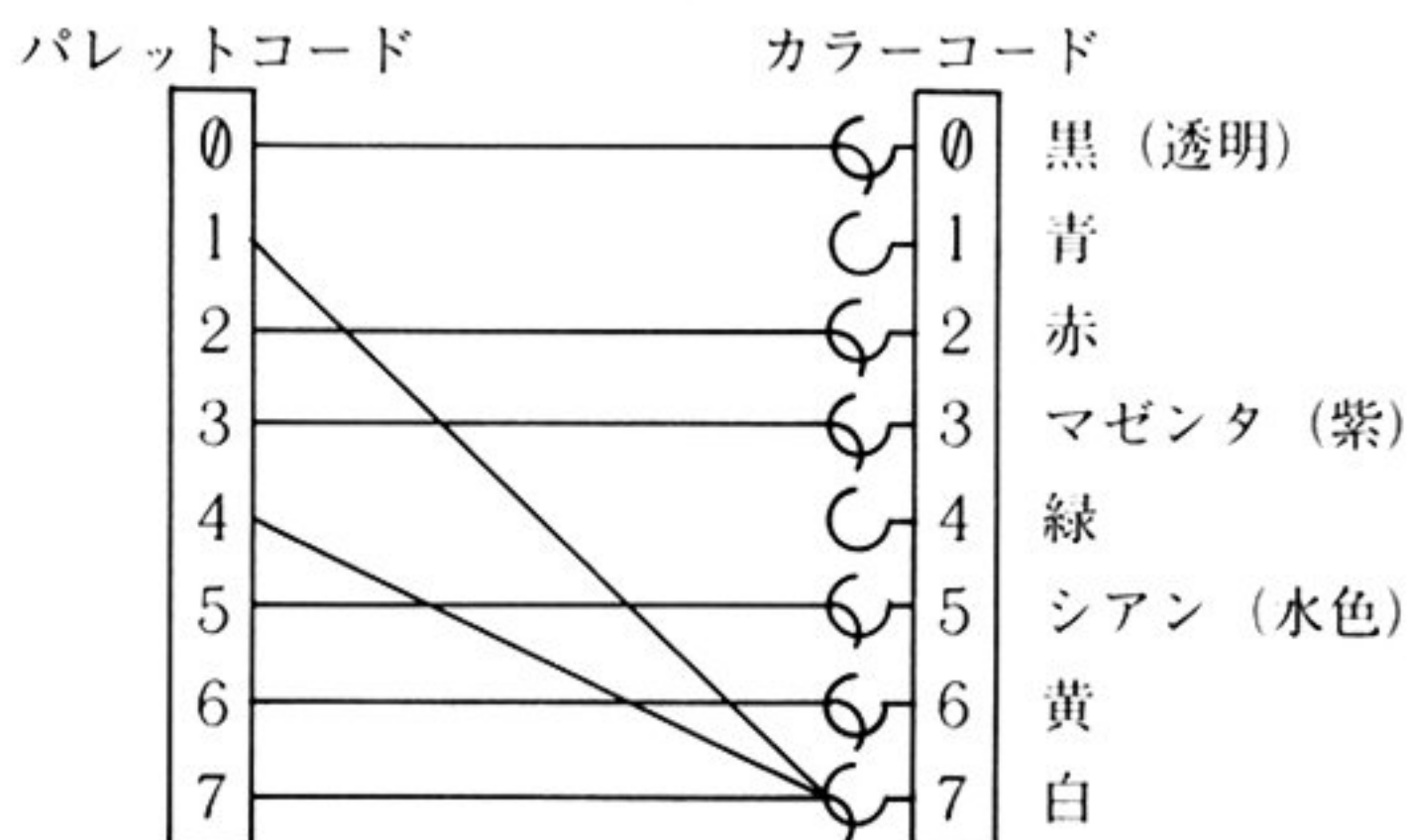
これに対してパレットコードも色を指定するコードで0～7の数字で表わされますが、カラーコードのように数字に対する色がきまっていませんので、自由に色を設定することができます。

パレットコードの色の設定はP A L E Tステートメントを使って行ないます。

次の図はパレットコードの設定の概念を示すものです。



B A S I Cが起動した最初の状態は上のようにパレットコード=カラーコードとなっていますが、次のようにフックをかけなおすと、パレットコードの1と4が白として使えるようになります。



## 5

## 中間色コード

グラフィックステートメントのうち

LINEのBF (ボックスフィル)

PAINT

SYMBOL

の3つに対して、パレットコードを指定する代わりに「中間色コード」を指定することができます。

中間色コードは、

& H m n (または  $m \times 16 + n$ )      $m = 0 \sim 7$  のパレットコード

$n = 0 \sim F$  のコード

の16進数2けたで表わされ、mとnの2色を混合した中間色を出すことができます。

ただし、mとnが同じパレットコードの場合は、パレットコードmかnを単独で指定したときの色になります。また、黒と他の色との混合色

& H 0 0、& H 0 1、……、& H 0 7



はカラーコードの

0、1、……、7

そのもので中間色を出すことができないので、黒と他の色との混合色は、

&H 0 8、&H 0 9、……、&H 0 F

で代用しています。

### <&H m n と &H n m の相違点>

&H 1 2 と &H 2 1 はパレットコード1（初期状態が青）とパレットコード2（初期状態が赤）を混合した同じ中間色（初期状態で紫）を表わしますが、グラフィック画面上のドット構成は次のように全く反転しています。

&H 1 2		&H 2 1	
(0, 0)		(0, 0)	
1	2	2	1
2	1	1	2
1	2	2	1
2	1	1	2

1 : パレットコード1でセット

2 : パレットコード2でセット

### <中間色一覧表>

中間色コードと表示される色との関係を次の表にまとめます。実際の色を確認する場合は表の下のプログラムを実行してください。

中間色コード &H m n または m\*16+n

n	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
m	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	黒 (0)	青 (1)	赤 (2)	マゼンタ (3)	緑 (4)	シアン (5)	黄 (6)	白 (7)	黒 (8)	ダーク ブルー (9)	茶 (10)	紫 (11)	深緑 (12)	青緑 (13)	黄土色 (14)	灰色 (15)
1	ダーク ブルー (16)	青 (17)	紫 (18)	青紫 (19)	青緑 (20)	空色 (21)	灰色 (22)	薄紫 (23)	<ul style="list-style-type: none"> <li>色は、初期状態のパレットコードによって出されるもの。</li> <li>( ) 内は10進数表現。</li> </ul>							
2	茶 (32)	紫 (33)	赤 (34)	ピンク (35)	黄土色 (36)	灰色 (37)	褐色 (38)	はだ色 (39)								
3	紫 (48)	青紫 (49)	ピンク (50)	マゼンタ (51)	灰色 (52)	薄紫 (53)	はだ色 (54)	薄ふじ色 (55)								
4	深緑 (64)	青緑 (65)	黄土色 (66)	灰色 (67)	緑 (68)	緑青 (69)	黄緑 (70)	ホワイト グリーン (71)								
5	青緑 (80)	空色 (81)	灰色 (82)	薄紫 (83)	緑青 (84)	シアン (85)	ホワイト グリーン (86)	水色 (87)								
6	黄土色 (96)	灰色 (97)	褐色 (98)	はだ色 (99)	黄緑 (100)	ホワイト グリーン (101)	黄 (102)	クリーム (103)								
7	灰色 (112)	薄紫 (113)	はだ色 (114)	薄ふじ色 (115)	ホワイト グリーン (116)	水色 (117)	クリーム (118)	白 (119)								

100 '中間色コード

110 INIT:WIDTH 80,12,0:CLS4

120 CSIZE2:PRINT#0," 中間色コード(&Hmn or m\*16+n)":CSIZE0:PRINT

130 PRINT " m":CHR\$(3H815F);;"n 0 1 2 3 4 5 6 7 8 9 A B C D E F"

;

140 FOR I=0 TO 7

150 LOCATE 3,I+3:PRINT I;

160 NEXT

170 FOR J=0 TO 7

180 FOR I=0 TO 15

190 LINE(67+I\*32,48+J\*16)-(96+I\*32,62+J\*16),PSET,J\*16+I,BF

200 NEXT

210 NEXT

220 LOCATE 0,0:END

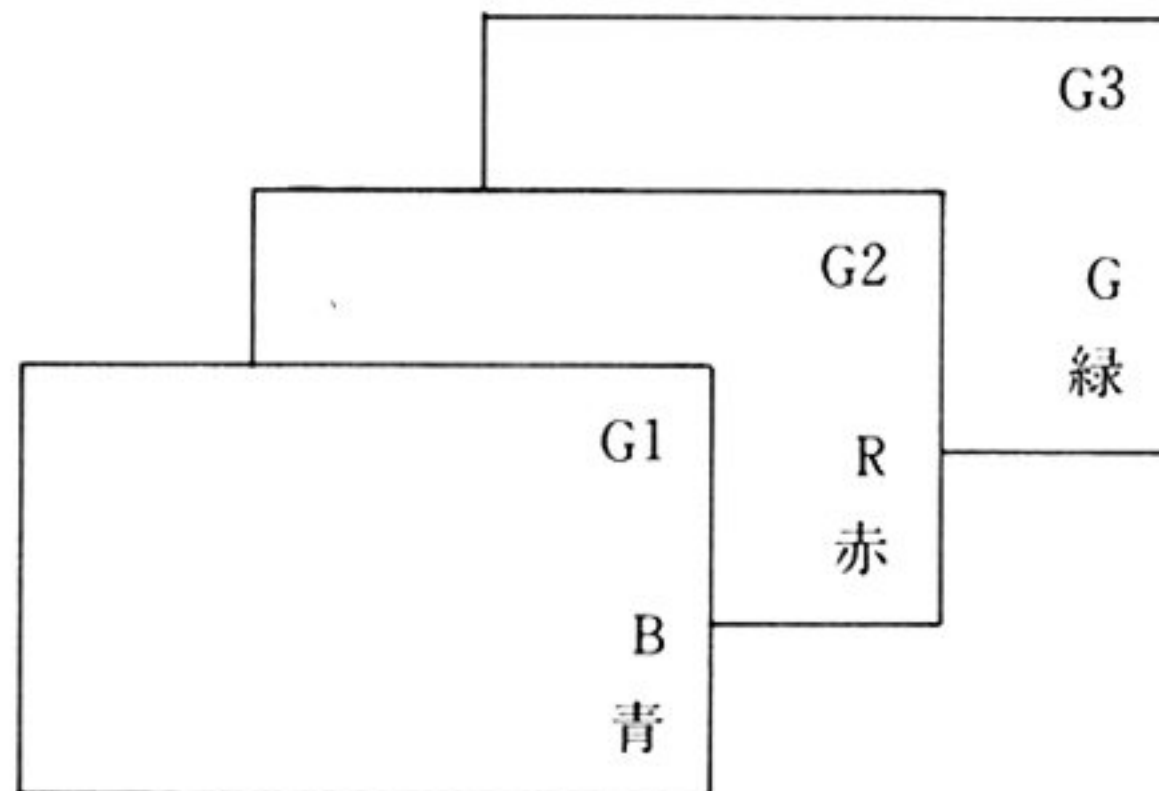
\*標準ディスプレイモードではグラフィック画面のドットが粗いので、中間色として見にくい部分があります。

## 6

## カラーモード

カラーモードとは、「SCREEN文の第3パラメータ（グラフィックモードの設定）が0となっていて、全グラフィック画面がアクセスできるモード」のことをいいます。

このモードのとき、グラフィック画面の各ドットに対して8色のうち任意の1色を指定することができます。



3枚のグラフィック画面を合成した画面で、8色のカラーを指定できる画面をカラー画面といいます。

画面の解像度の設定により、何ページかの独立したカラー画面を設定することができます。

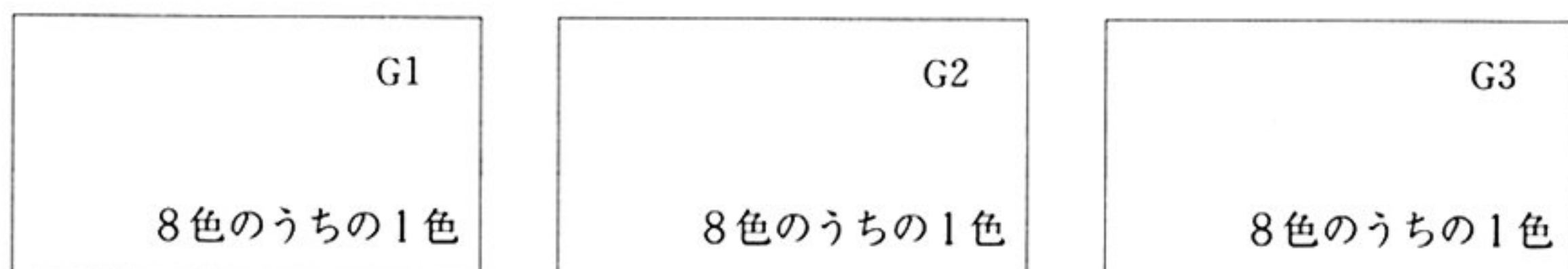
解 像 度	カラー画面のページ数
320×200、320×192	4 (PAGE 0 ~ PAGE3)
640×200、640×192	2 (PAGE 0 , PAGE1)
320×400、320×384	
640×400、640×384	1

カラー画面が複数ページであるときは、SCREENステートメントの第1パラメータと第2パラメータで表示するページと書き込むページを指定することができます。

## 7

## マルチページモード

マルチページモードとは、「SCREEN文の第3パラメータ（グラフィックモード）の指定が1、2、3のいずれかになっていて、G1、G2、G3の3枚の画面を各々単色画面としてアクセスできるモード」のことをいいます。このモードのとき、CANVASステートメントによって、各画面の色を任意の1色に設定することができます。



画面の解像度により、使用できる画面のページ数を設定することができます。

ステートメント	解像度	マルチ画面のページ数
WIDTH40, 25, 0, d	320×200	3×4
WIDTH40, 12, 0, d	320×192	
WIDTH80, 25, 0, d	640×200	3×2
WIDTH80, 12, 0, d	640×192	
WIDTH40, 25, 0, D	320×400	
WIDTH40, 12, 0, D	320×384	
WIDTH80, 25, 0, D	640×400	3×1
WIDTH80, 12, 0, D	640×384	

※ d：0、1、2（標準ディスプレイモードまたは高解像度ディスプレイモードに設定。）

D：0、2（高解像度ディスプレイモードに設定。D=0のとき本体の標準／高解像度切換スイッチは、HIGH側(■)にセットされていること。)

## 8

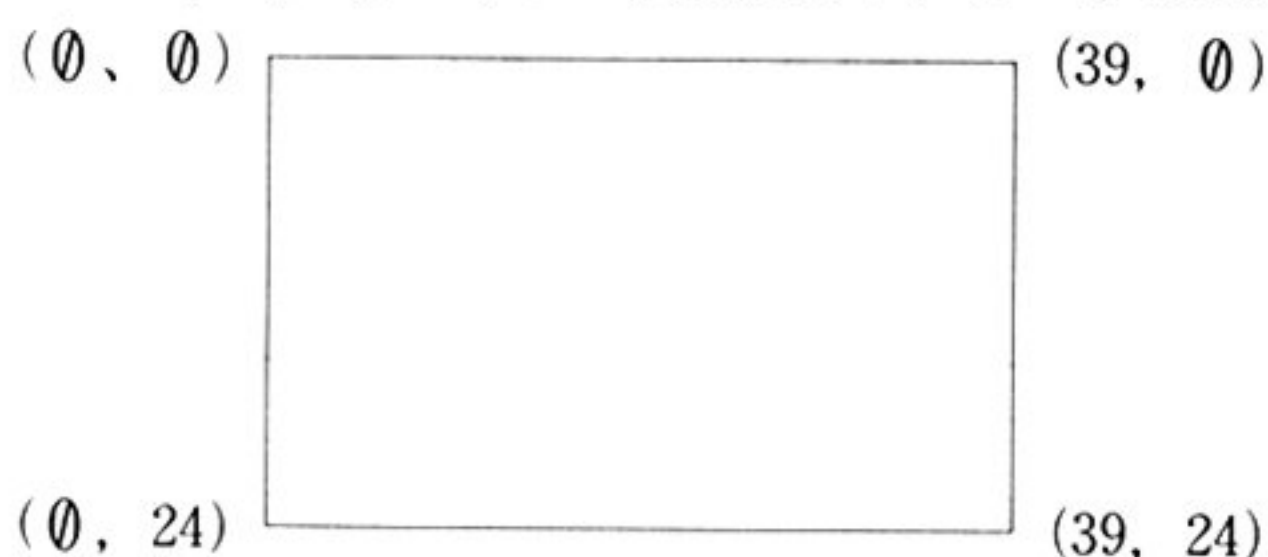
## ディスプレイ画面上の座標系

テキスト画面、グラフィック画面とも画面の左上が原点（0，0）であり、横方向がx軸、たて方向がy軸となっています。

### 8.1 テキスト座標系

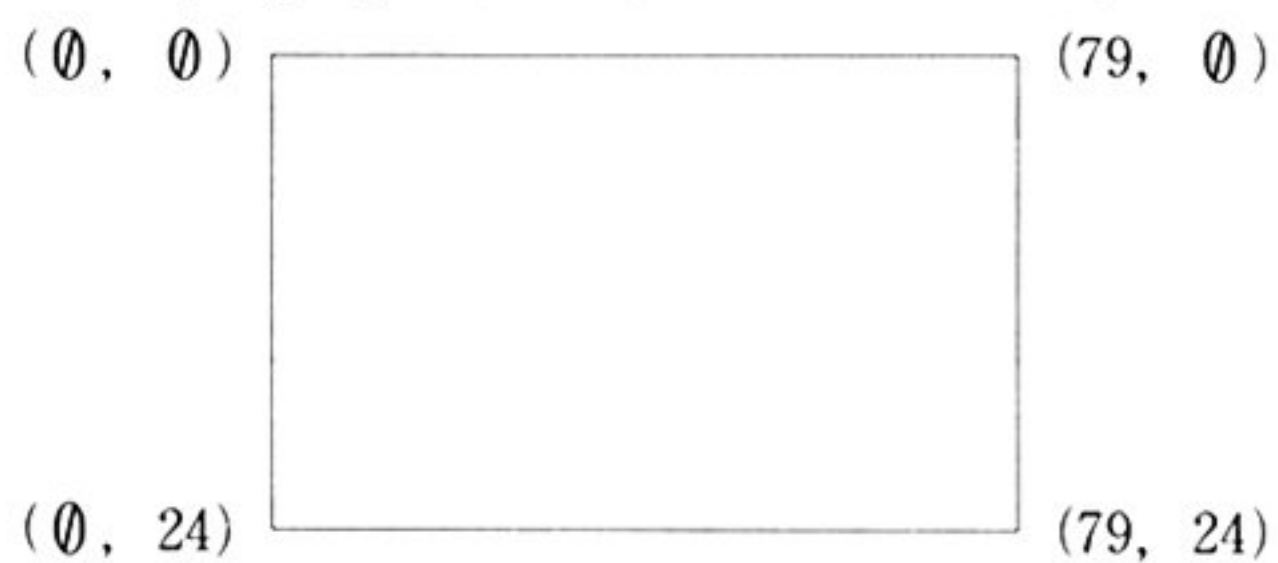
テキスト座標系は、テキスト画面において設定されている座標系で、表示文字数によって次の図のようになっています。

〈WIDTH40, 25, G, D（G=0または1、D=0または1または2）のとき〉

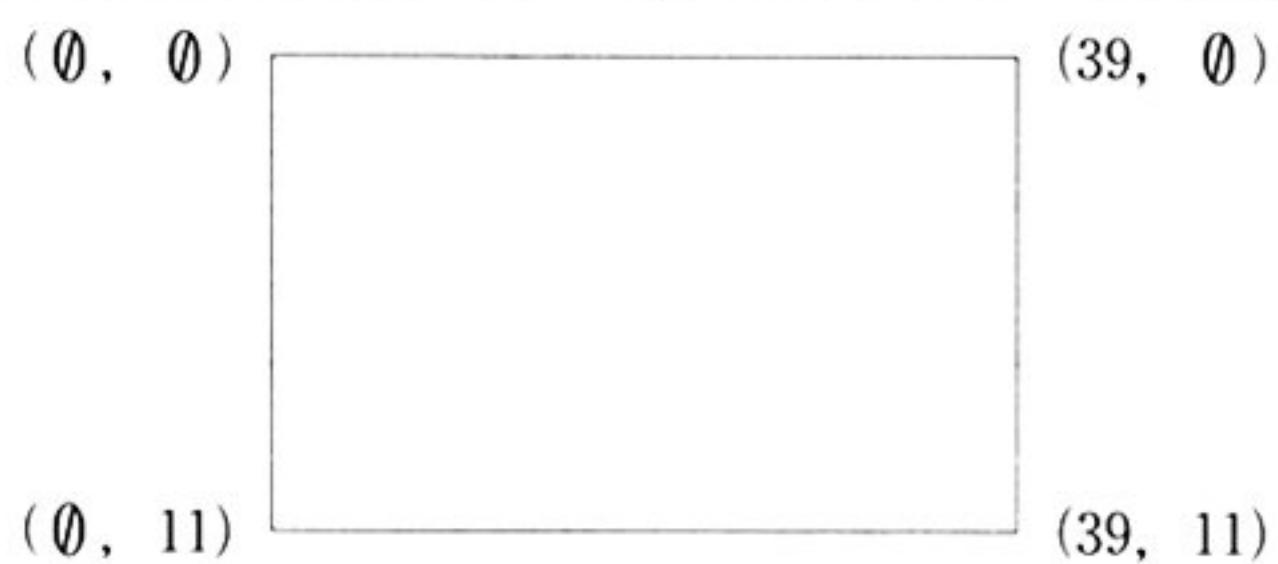




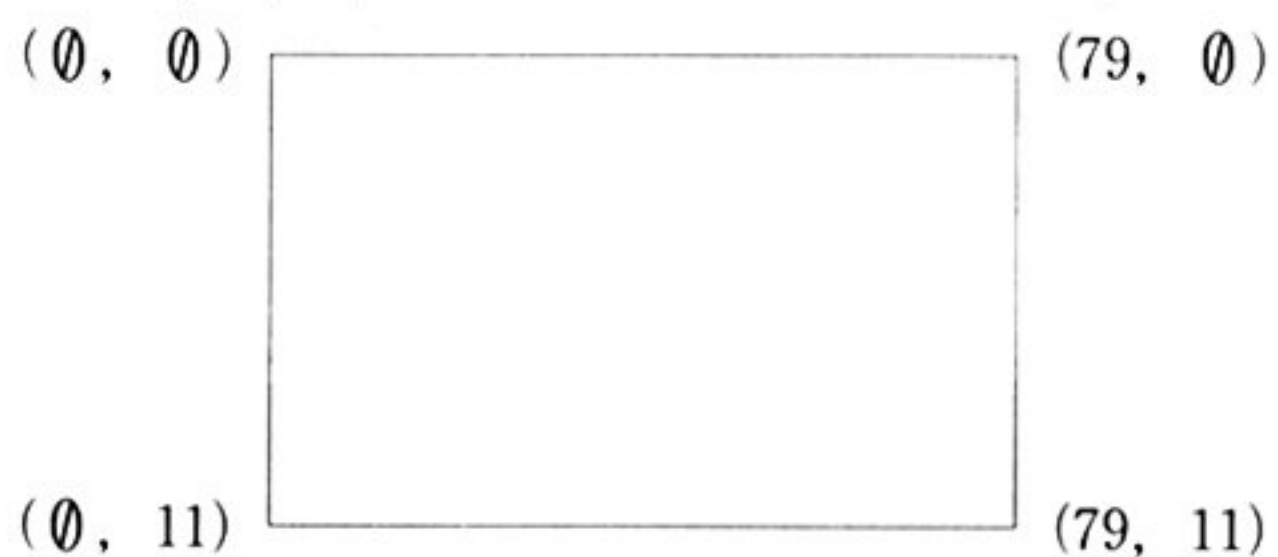
〈WIDTH80,25,G,D (G=0または1、D=0または1または2) のとき〉



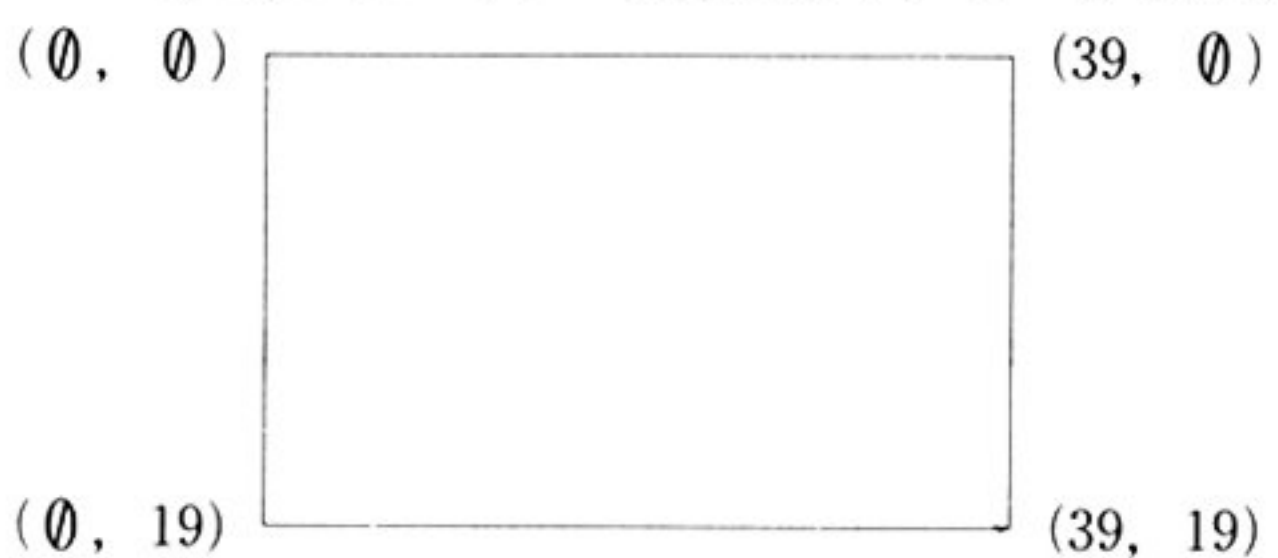
〈WIDTH40,12,G,D (G=0または1、D=0または1または2) のとき〉



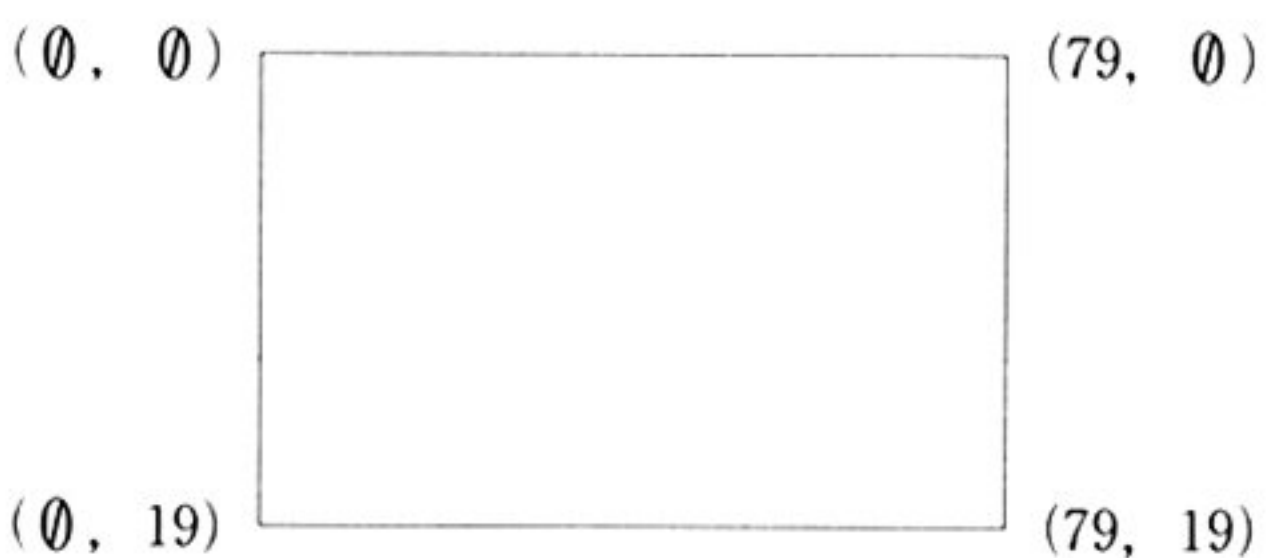
〈WIDTH80,12,G,D (G=0または1、D=0または1または2) のとき〉



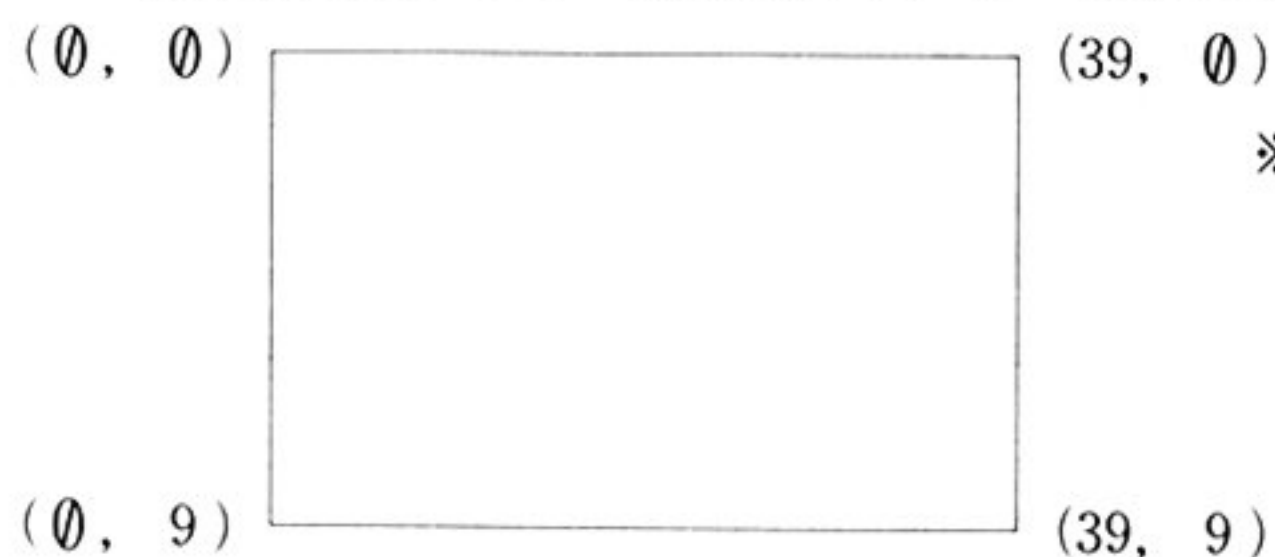
〈WIDTH40,20,G,D (G=0または1、D=0または1または2) のとき〉



〈WIDTH80,20,G,D (G=0または1、D=0または1または2) のとき〉

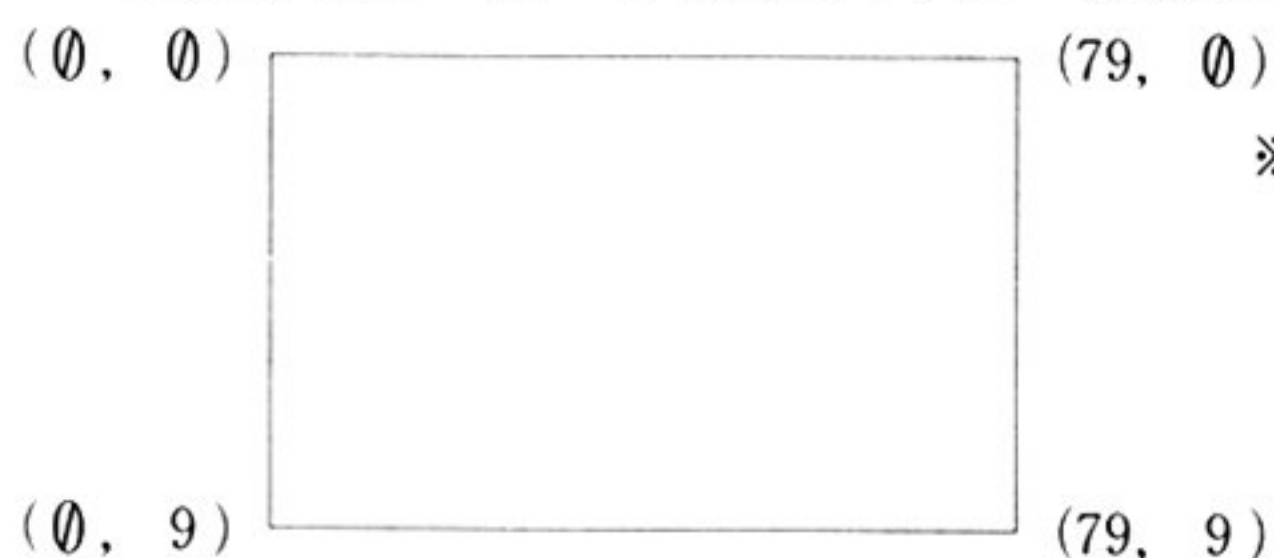


〈WIDTH40,10,G,D (G=0または1、D=0または1) のとき〉



※D=0のとき本体の標準／高解像度切換スイッチは標準ディスプレイモード(■)にセットされていること。

〈WIDTH80,10,G,D (G=0または1、D=0または1) のとき〉



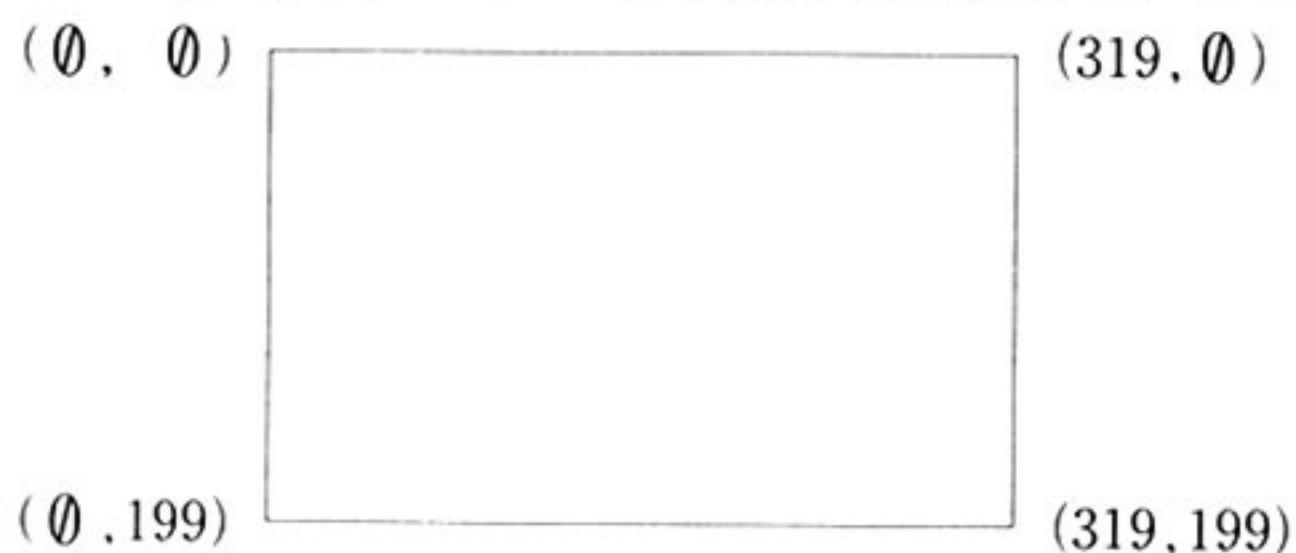
※D=0のとき本体の標準／高解像度切換スイッチは標準ディスプレイモード(■)にセットされていること。

CONSOLEおよびLOCATEステートメントは、前ページに示した座標の範囲で設定することができます。

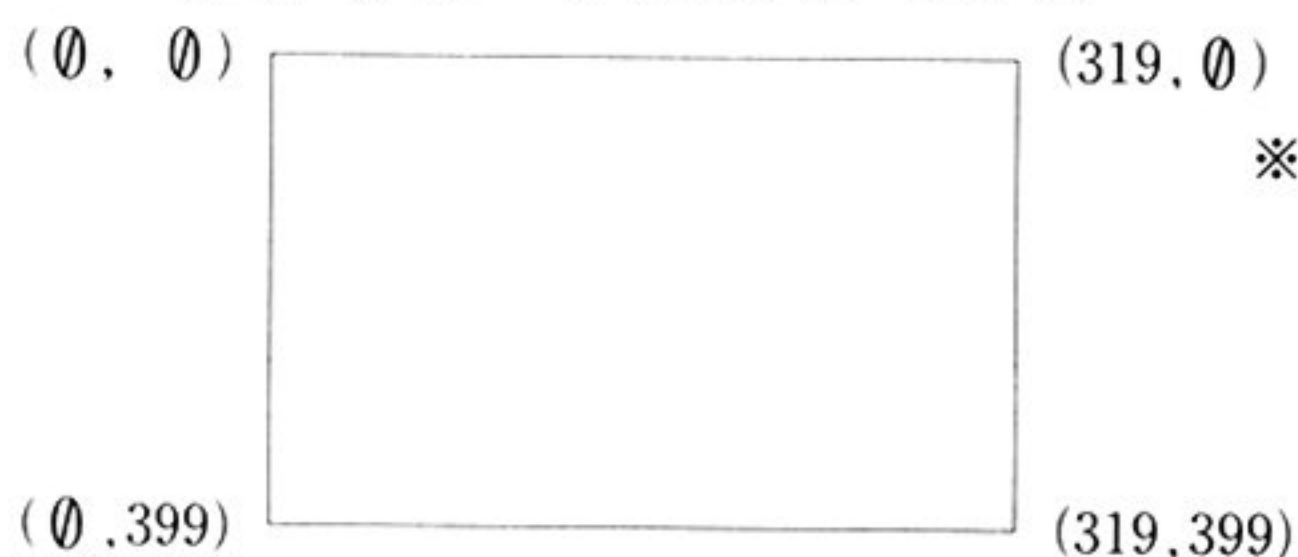
## 8.2 グラフィック座標系

グラフィック座標系は、グラフィック画面において設定されている座標系です。  
グラフィックの解像度によって、次のような座標の範囲が設定されます。

〈WIDTH40,25,0,D (D=0または1または2) のとき〉

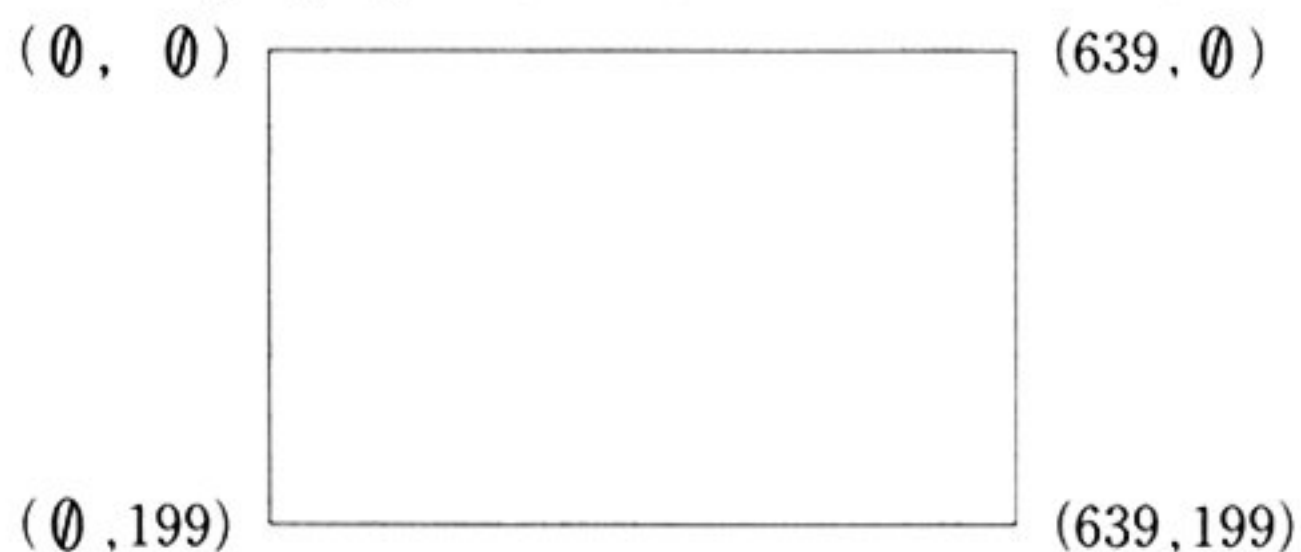


〈WIDTH40,25,1,(D=0または2) のとき〉

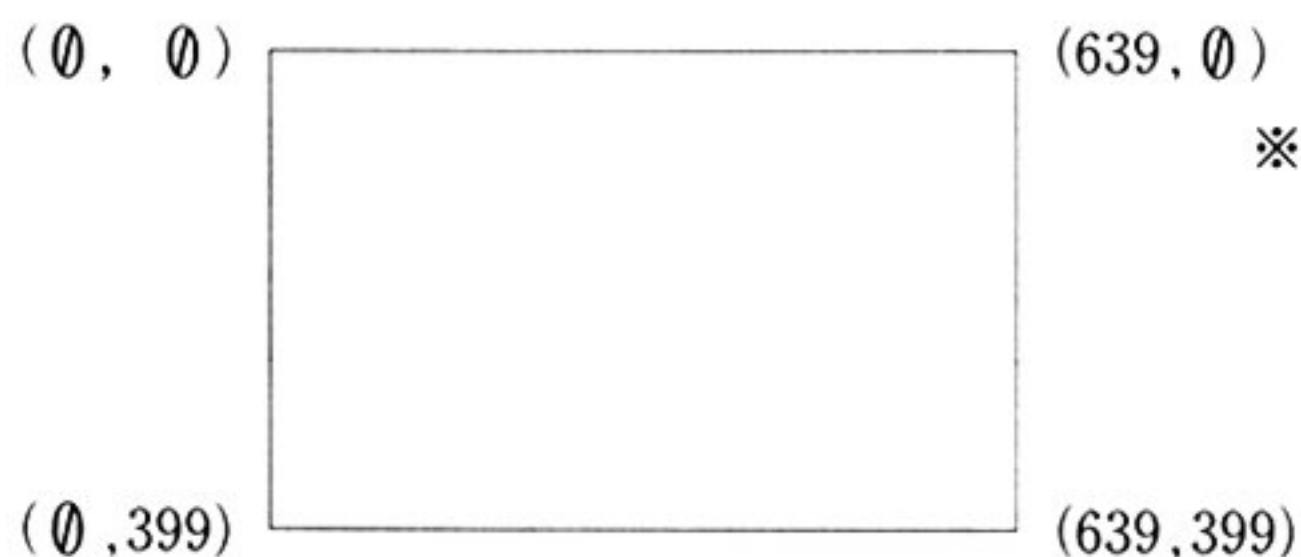


※D=0のとき本体の標準／高解像度切換スイッチは高解像度ディスプレイモード(■)にセットされていること。

〈WIDTH80,25,0,D (D=0または1または2) のとき〉

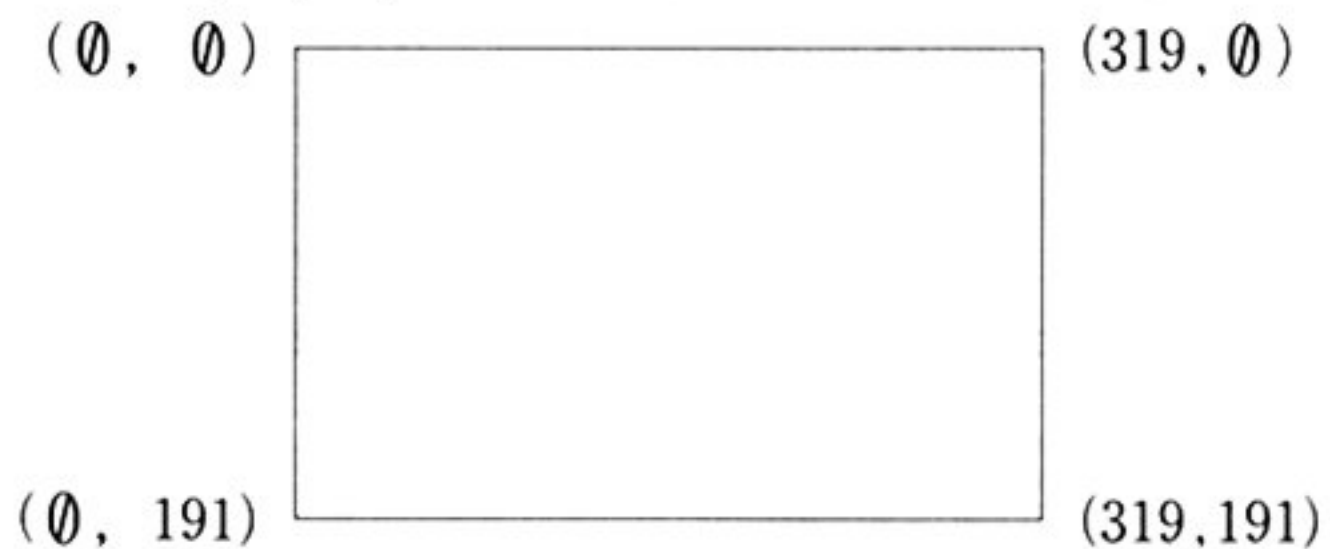


〈WIDTH80,25,1,D (D=0または2) のとき〉

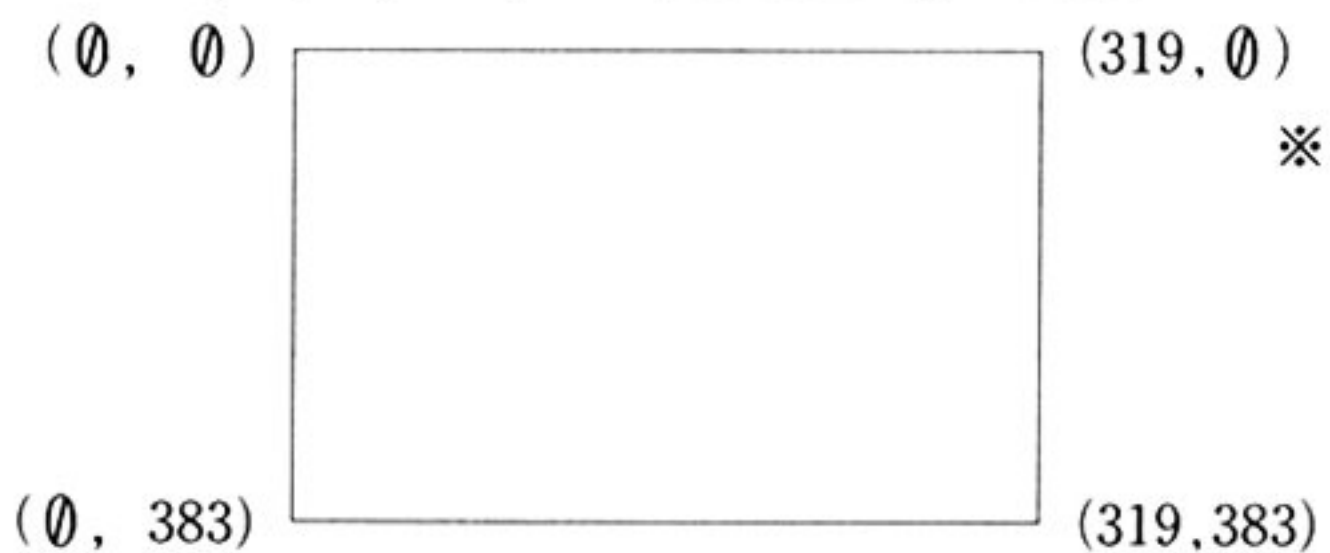


※D=0のとき本体の標準／高解像度切  
換スイッチは高解像度ディスプレイモード  
(■) にセットされていること。

〈WIDTH40,12,0,D (D=0または1または2) のとき〉

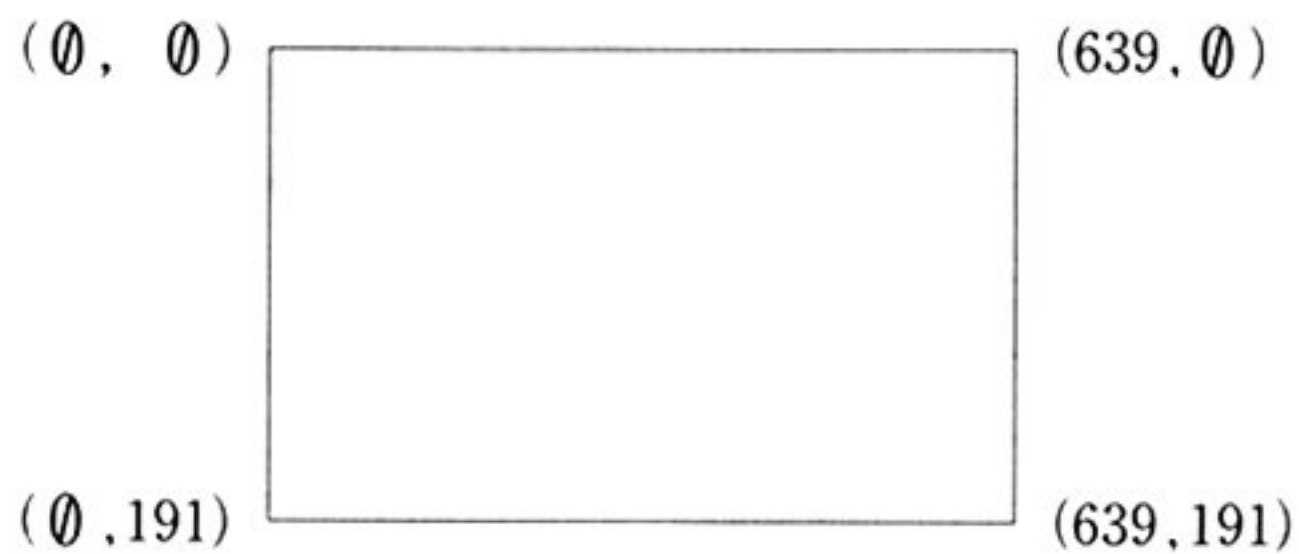


〈WIDTH40,12,1,D (D=0または2) のとき〉



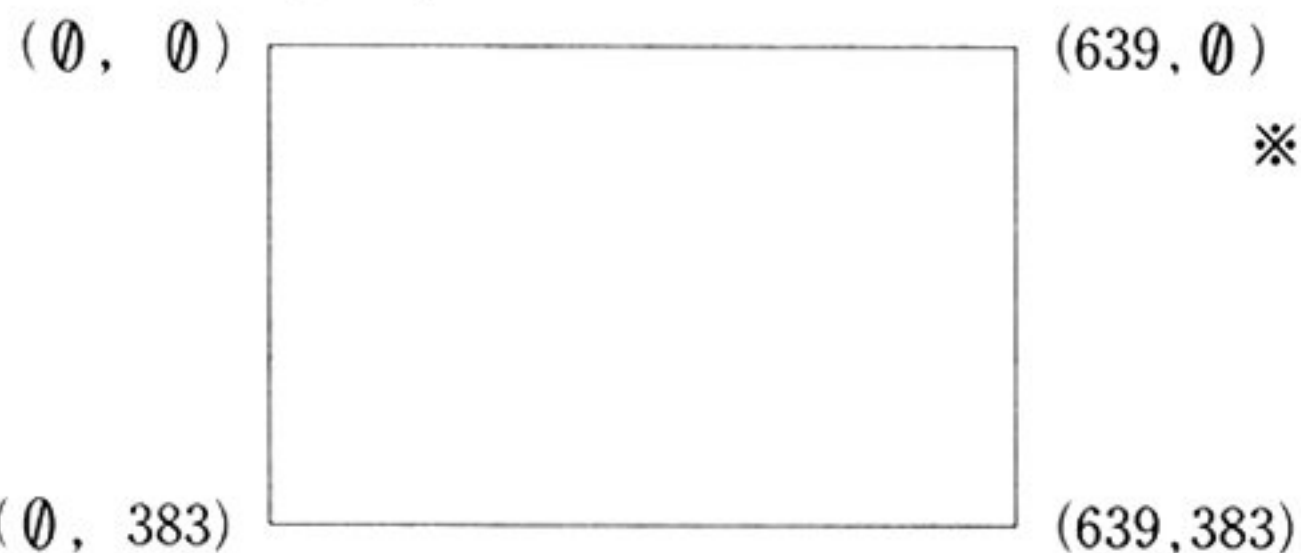
※D=0のとき本体の標準／高解像度切  
換スイッチは高解像度ディスプレイモード  
(■) にセットされていること。

〈WIDTH80,12,0,D (D=0または1または2) のとき〉





〈WIDTH80, 12, 1, D (D=0または2) のとき〉



※D=0のとき本体の標準／高解像度切換スイッチは高解像度ディスプレイモード(■)にセットされていること。

※WIDTH40, 20, G, D (G=0または1、D=0または1または2)、  
WIDTH80, 20, G, D (G=0または1、D=0または1または2)、  
WIDTH40, 10, G, D (G=0または1、D=0または1)、  
WIDTH80, 10, G, D (G=0または1、D=0または1)

のときはグラフィック画面が使用できません。したがって、グラフィック座標系は設定されません。

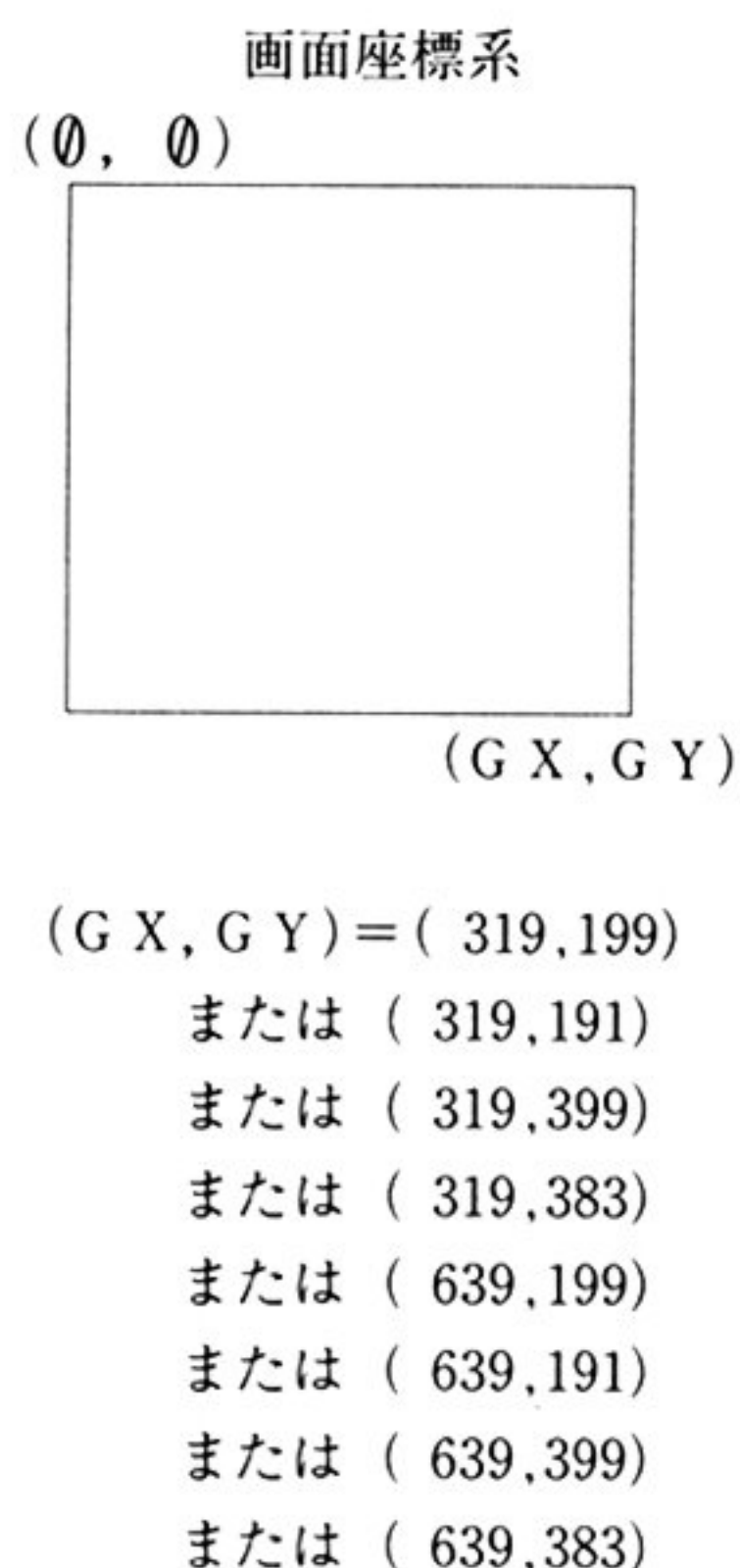
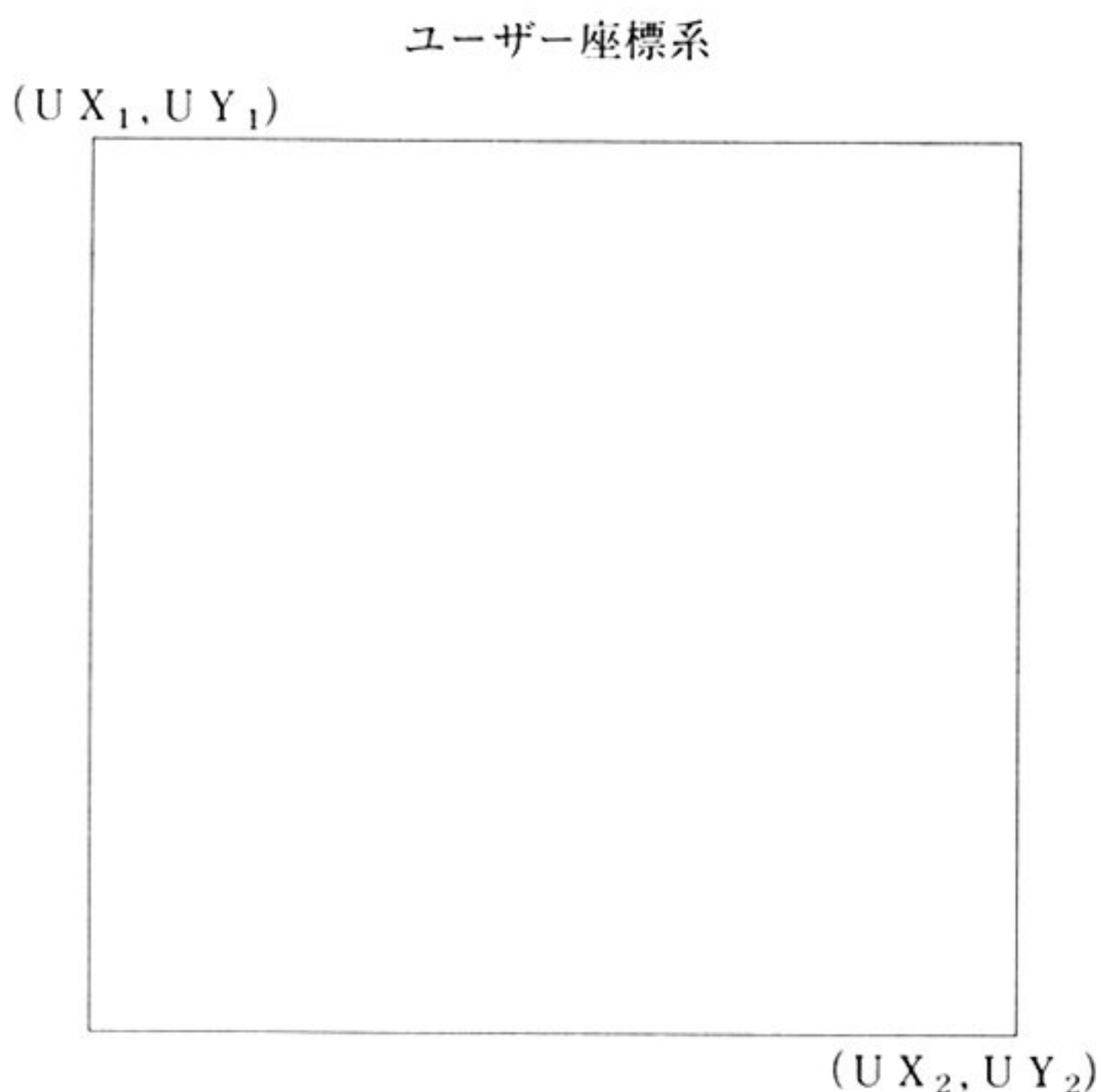
### 8.3 ユーザー座標系と画面座標系

ディスプレイ画面は、テキスト画面とグラフィック画面が重なって表示され、テキスト画面にはテキスト座標系、グラフィック画面にはグラフィック座標系が設定されています。このうちグラフィック座標系は画面座標系とユーザー座標系という2つの座標系をもっています。

画面座標系とは、グラフィック座標系で説明した座標に一致するもので、画面上の1ドットが座標単位の1に対応します。

ユーザー座標系とは、BASICのWINDOWステートメント(『BASICリファレンスマニュアル』の2.8.2 WINDOW参照)によってユーザーが指定した座標系で、たて方向、横方向とも $-1.7014118E+38 \sim 1.7014118E+38$  (指数部-38~+38)の単精度の範囲で指定できます。PSET, PRESET, LINE, POLY, CIRCLE, PAINTのグラフィックステートメント、およびPOINT関数はこのユーザー座標で使

用されます。下の図は、ユーザー座標と画面座標系です。




$$(U X_1, U Y_1) = (-1.7014118E+38, -1.7014118E+38)$$

$$(U X_2, U Y_2) = (+1.7014118E+38, +1.7014118E+38)$$

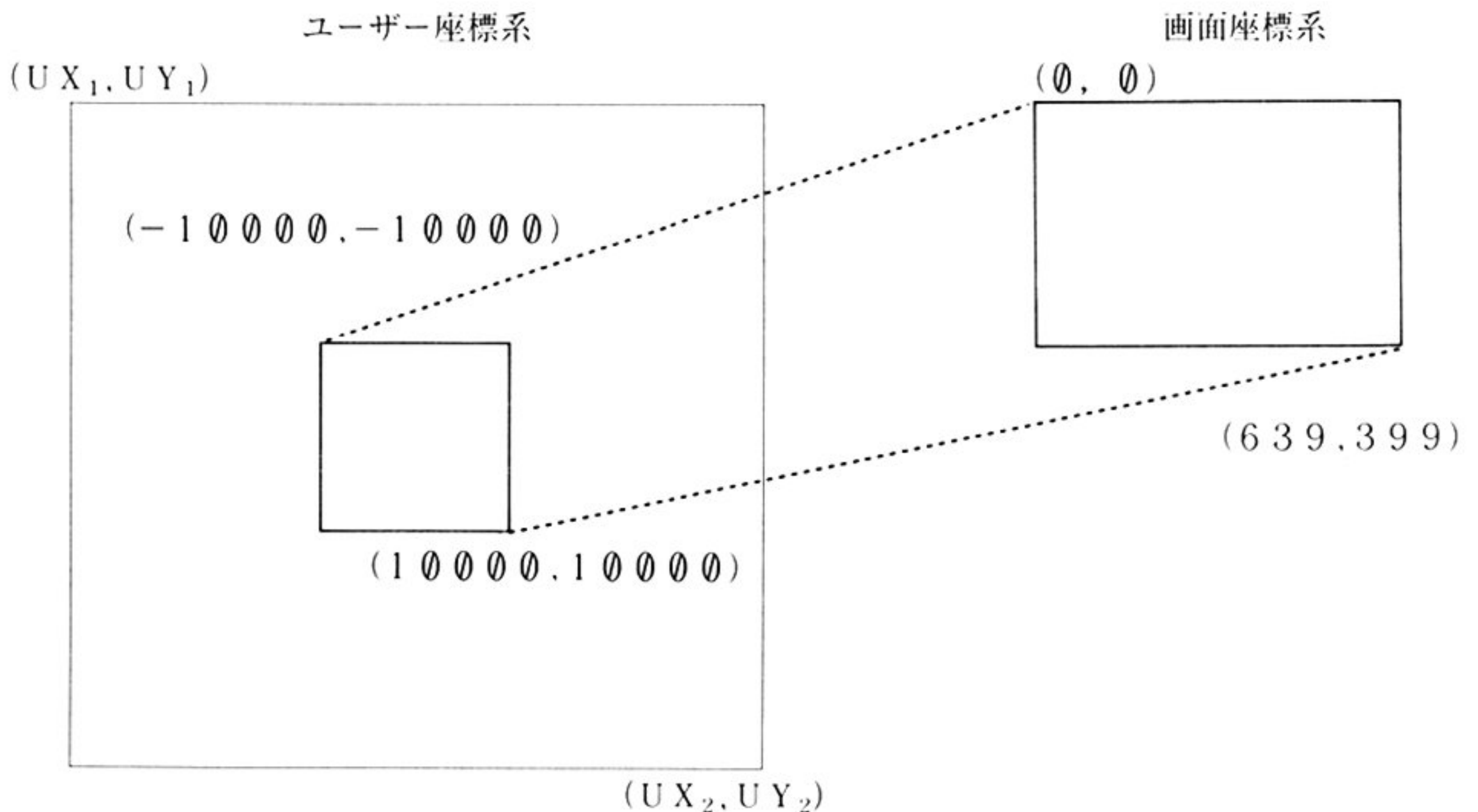
画面座標系は実際に目で見ることはできますが、ユーザー座標系の広大な座標平面は目で見ることはできません。

したがって、BASICのWINDOWステートメントを使って、ユーザー座標系の座標平面中の任意の領域を画面座標系の平面上に割り当てなければなりません。


たとえば、WIDTH 80, 25, 1, 2が設定されているとき、

**WINDOW (0, 0) - (639, 399), (-100000, -100000) - (100000, 100000)** 

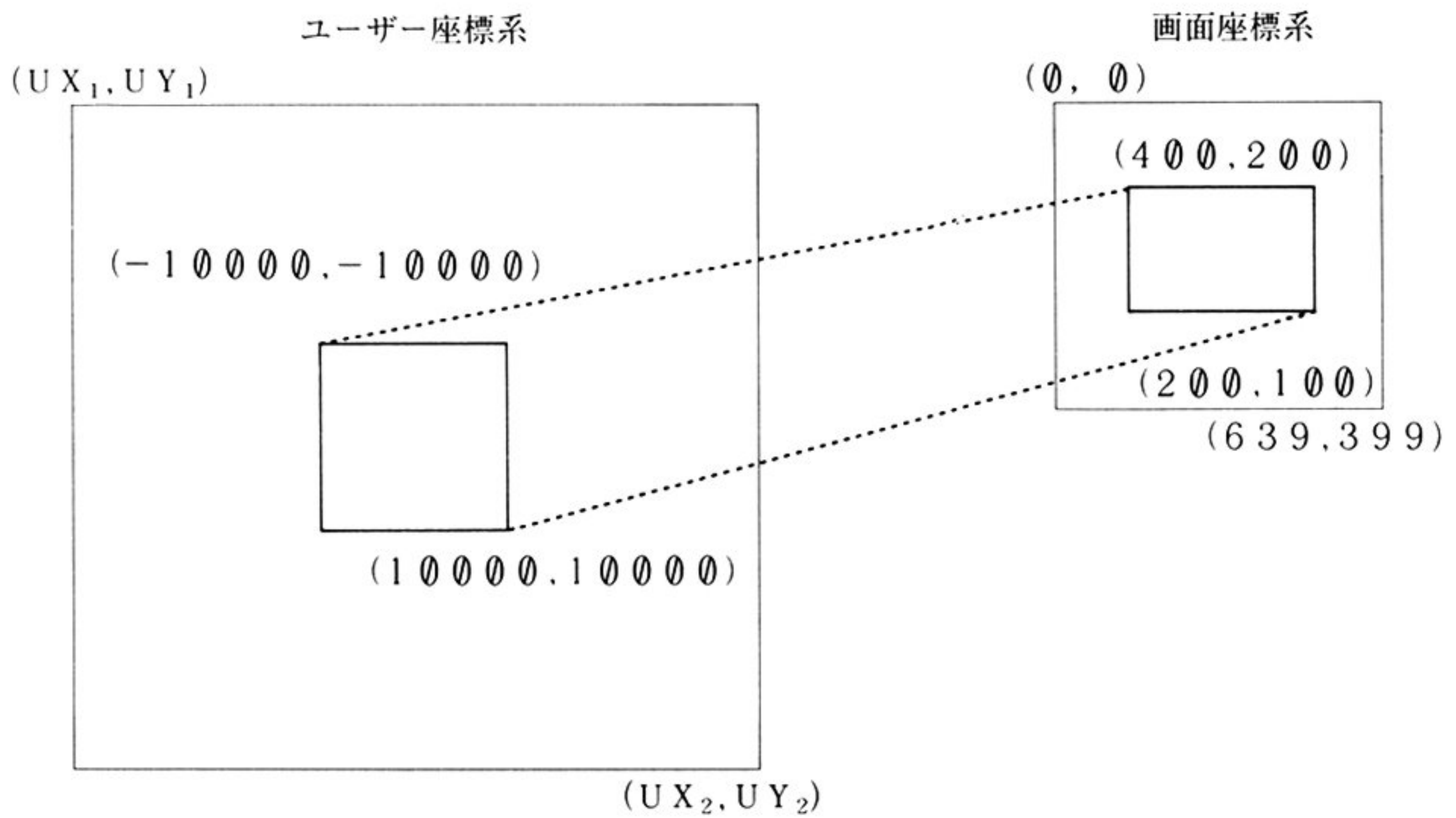
と実行すると、次のようにユーザー座標系の(-100000, -100000)から(100000, 100000)を対角線とする長方形の領域を画面座標系全域(グラフィック画面全体)に表示することができます。



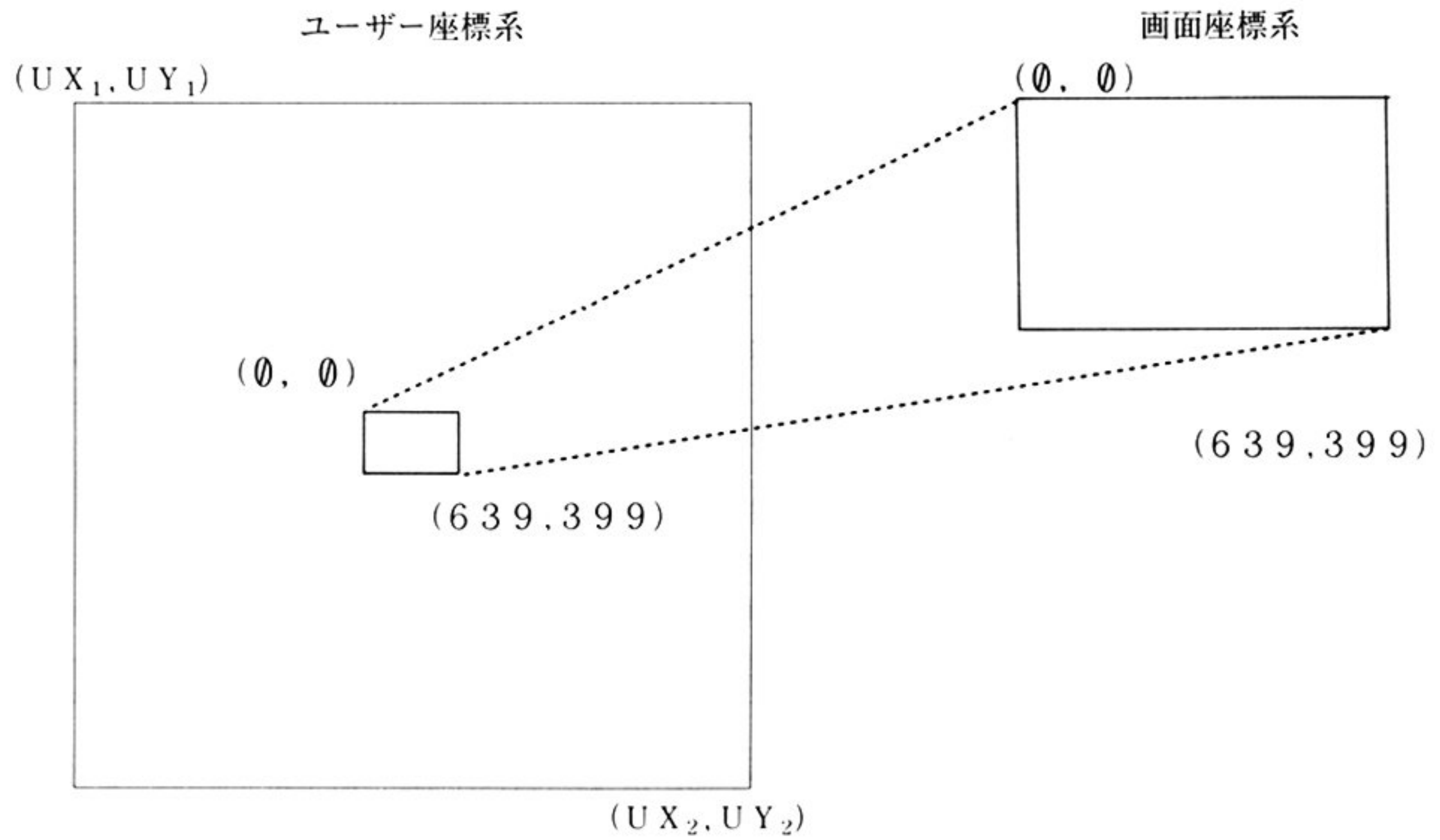
また、

**WINDOW (200, 100) - (400, 200), (-100000, -100000) - (100000, 100000)** 

を実行すると、ユーザー座標系の(-100000, -100000)から(100000, 100000)を対角線とする長方形の領域を画面座標系の(200, 100)から(400, 200)を対角線とする長方形の領域に表示することができます。



後ろのパラメータを省略して、WINDOWのみを実行すると、画面座標系の範囲と同じ領域が自動的に指定されます。





# 3章 テキスト

## 3章

キーボードから入力された文字やプログラムのリストはディスプレイ画面に表示されます。このとき、もしディスプレイ画面に円や線などのグラフィックが描かれていれば、その上に重なって文字が表示されます。ここで画面消去の命令 `CLS` を入力すると文字のみが消去され、グラフィックはそのまま残って表示されています。グラフィックを描く画面と文字を表示する画面は独立しており、それぞれグラフィック画面とテキスト画面といいます。

### 1

## テキスト画面

テキスト画面にはアルファベット、数字、漢字、カタカナ、ひらがな、セミグラフィック文字などが書かれます。

`WIDTH` 命令は画面に表示する文字数を設定します。さらに、`CONSOLE` 命令を使用すると、表示画面をテキストの命令が有効な領域と無効な領域に分けることができます。

**CONSOLE** たて方向の表示開始行，たて方向の表示行数，横方向の表示開始行，横方向の表示桁数

たとえば、横方向の最初の桁から 20 桁とたて方向の最初の行から 10 行の領域をテキストの命令が有効な領域に設定するには、

`CONSOLE 0, 10, 0, 20` を入力します。

すると、`LIST` や `FILES` 命令を実行した場合、この設定された領域（20 文字×10 行）内でリストやファイル名が表示され、それ以外の領域には表示されません。このようにして、テキスト表示領域内で、表示文字の変化可能領域と変化不可能領域とに分けることができます。この画面分割を解除するには、再度 `CONSOLE` 命令を実行して表示画面全体を指定するかまたは `CTRL` + `D` キーと押してください。

また、テキスト画面の任意の位置に文字を表示したりする場合には `LOCATE` 命令を使用します。

**LOCATE** 横方向の位置，たて方向の位置

たとえば、横方向 10 桁目、たて方向 10 行目の位置から `ABC` という文字を表示させるには

`LOCATE 10, 10:PRINT "ABC"` を入力します。

## 1.1 ファンクションキーの内容表示

テキスト画面の最下位は、ファンクションキーの内容表示に使用することができます。

ただし、表示画面の最下行は、CONSOLEステートメントによりテキストへの命令が無効領域に設定されている必要があります。

ファンクションキーの表示は `KEY LIST` という命令を使って行ないます。

```
KEY LIST 0
```

と入力するとファンクションキーの内容は表示されず、

```
KEY LIST 1
```

と入力すると表示されます。

## 1.2 漢字とセミグラフィックパターン

本機は、漢字をテキスト画面に表示します。漢字を表示する場合画面モードを漢字表示のモードに設定しておく必要があります。この設定を行なうには、`KMODE` という命令を使います。

```
KMODE 0 }  
      1 }
```

```
KMODE 1
```

と入力すると漢字表示モードになります。

ただし、標準ディスプレイモードでたて方向の行数が25行または20行に設定されている場合には、漢字は表示できません。

このモードで「会社」という文字を表示するには

```
PRINT "会社"
```

あるいは

```
PRINT CHR$(&J3271, &J3C52)
```

と入力します。また、

```
PRINT CHR$(&H89EF, &H8ED0)
```

と入力しても

会社

と表示されます。

漢字コードは内部では2バイトで表現されており、最初の1バイトが&H80～&H9F もしくは、&HE0～&HFFではじまるコードとなっています。&H80～&H9F、&HE0～&HFFのコードはセミグラフィックパターンのコードと重複しています。したがって漢字とセミグラフィックパターンとは同時に表示できません。そこで漢字の代りにセミグラフィックパターンを表示するには

```
KMODE 0
```

と入力します。ここで

```
PRINT CHR$(&H89EF)
```

と入力してみてください。今度は「会」という文字ではなく ■□というセミグラフィックパターンが表示されます。



テキストは最大80文字×25行の2000文字を表示することができますが、1文字単位で、色を指定することができます。また、反転や点滅も同様に1文字単位で行なえます。

この色や反転、点滅のことを**文字の属性**といいます。

そのほか文字の属性には倍文字、アンダーライン、ROM/RAMキャラクタジェネレータがあります。

文字の属性は英数字、カナ、漢字すべてに指定できます。

## 2.1 色の指定

文字の色はCOLORという命令で行なえます。

**COLOR 色の番号**

0：黒、1：青、2：赤、3：マゼンタ、4：緑、5：シアン、6：黄、7：白  
 とすると、この命令が実行されてから以後に入力される文字は指定された色になります。

また、**CTRL** キーを押しながらテンキーの**0**～**7**を押すことによっても文字の色指定ができます。

## 2.2 反転文字

文字は標準状態では指定された色で表示されていますが、反転モードにして入力すると、文字の色が補色になり白い四角で囲まれた状態になります。

補色は

青↔黄

赤↔シアン

緑↔マゼンタ

白↔黒

となります。



**CREV { 0または省略  
1 }**

反転モードにするには

**CREV 1** 

と入力します。

また

**CREV 0**  もしくは **CREV** 

とすると標準モードにもどります。

また、**CTRL** キーを押しながらテンキーの**-**キーを押すことによっても反転モード、標準モードの切り換えができます。



## 2.3 点滅文字

点滅モードにすると、入力された文字は反転の状態と標準の状態を繰り返し表示します。



**CFLASH** { 0または省略  
                  1 }

点滅モードにするには

**CFLASH 1** 

とします。

標準モードにするには

**CFLASH 0**  または **CFLASH** 

とします。また **CTRL** キーを押しながらテンキーの **\*** キーを押すことによっても点滅モード、標準モードの切り換えができます。

## 2.4 倍文字

本機では4種類の文字を扱うことができます。これは、グラフィックで1ドットずつ文字を作るのではなく **PRINT** 文で使えます。この4種類を区別して使いわけける命令は **CSIZE** で、書式は次のとおりです。

**CSIZE n**

nには0、1、2、3のいずれかの数字を代入します。数字を代入した書式の内容は次のとおりです。

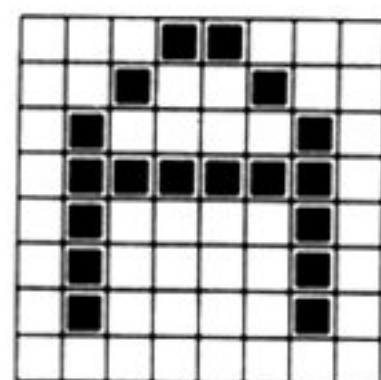
**CSIZE 0**：標準の8×8ドットの文字です。

**CSIZE 1**：縦方向に2倍の文字です。

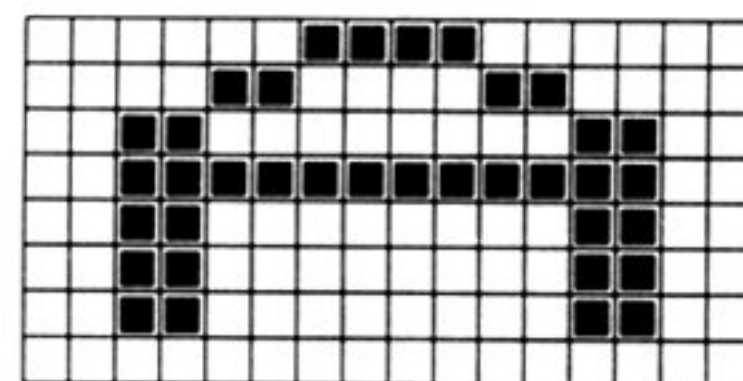
**CSIZE 2**：横方向に2倍の文字です。

**CSIZE 3**：縦方向、横方向ともに2倍の文字です。

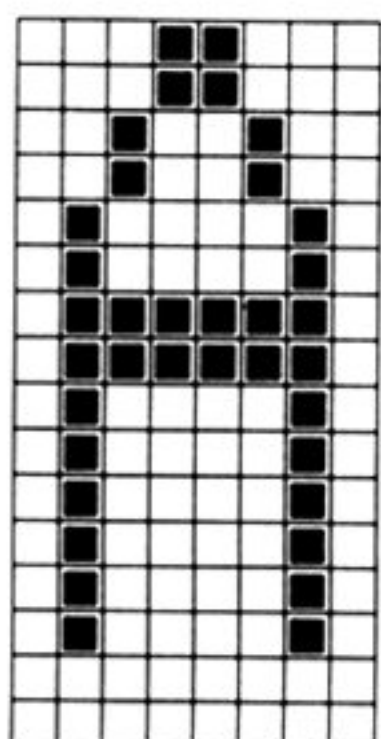
CSIZE 0



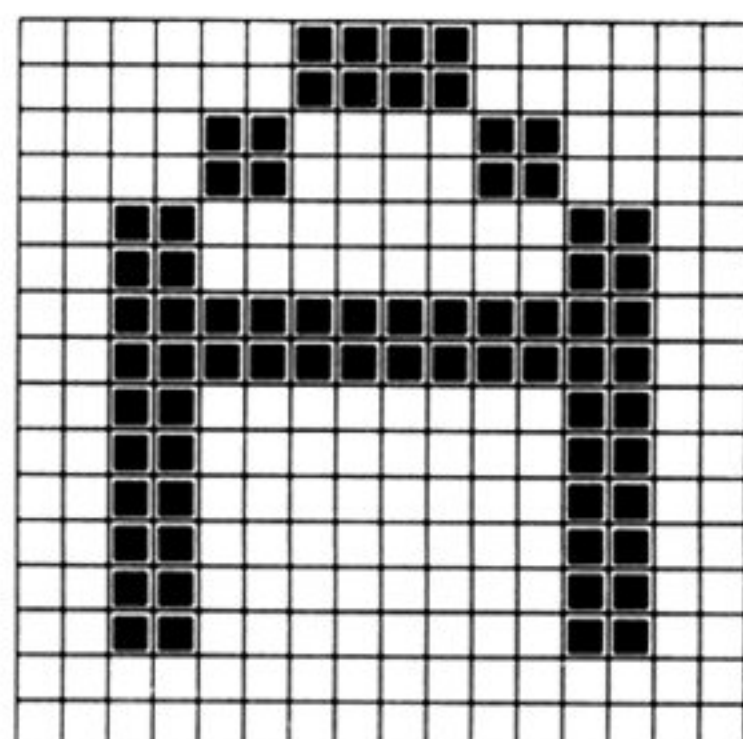
CSIZE 2



CSIZE 1



CSIZE 3



倍文字を書く場合、まずC S I Z E文でサイズを指定します。次にP R I N T文を使いますが、ここで使うP R I N T文には# 0という記号を入れ書かせようとする文字列を続けて入れます。

P R I N T # 0, " [文字列] "

たとえば 10 C S I Z E 2  
20 P R I N T # 0, "ABC" とすると「A B C」が横2倍の文字で書かれます。

文字を表示する座標の指定はL O C A T E命令を使用します。文字の表示座標を指定する場合、次の点に注意してください。

- ① C S I Z E 1 (たて2倍) または、C S I Z E 3 (たて、横2倍) を指定した場合のたて方向の座標に関する注意点。

表示する文字より上にある他の文字との間隔は偶数行 (0、2、4…行) あけなければなりません。もし上に何も文字がない場合には、偶数座標にしか表示できません。奇数座標に表示したい場合には、その座標より上方向に偶数行あけたスペースを挿入することにより可能となります。また、たて2倍あるいはたて、横2倍文字が表示されている2行内に、標準文字および横2倍文字を表示することはできません。

- ② C S I Z E 2 (横2倍) または、C S I Z E 3 (たて、横2倍) を指定した場合の横方向の座標に関する注意点。

横方向の座標を1、3、5、…の奇数座標に指定しても、指定した奇数座標に1を加えた2、4、6、…の偶数座標に文字は表示されます。よって、横2倍文字またはたて、横2倍文字を表示する場合の横方向の座標は0、2、4…の偶数座標を指定してください。

### サンプル プログラム

#### ちょっとコマーシャル

文字の大きさを変えて "SHARP PERSONAL COMPUTER X-1" と表示するものです。

```
50 REM C S I Z E ヲ ツカフ
100 WIDTH 40 : INIT
110 C S I Z E 3                      ←縦横2倍文字にします
120 CLS
200 LOCATE 5, 8
210 COLOR 4
220 P R I N T # 0, "SHARP "          ←縦横2倍文字で "SHARP" を表示
240 LOCATE 10, 12
250 COLOR 2
260 C S I Z E 1                      ←縦2倍文字にします
270 P R I N T # 0, "PERSONAL COMPUTER" ←縦2倍文字で "PERSONAL COMPUTER" を表示
300 LOCATE 20, 16
310 COLOR 6
320 C S I Z E 2                      ←横2倍文字にします
330 P R I N T # 0, "X-1 "           ←横2倍文字で "X-1" を表示
400 END
```



## 2.5 ROMCGとRAMCG

テキスト画面に表示される文字はキャラクタジェネレータと呼ばれる部分から呼び出されたパターンが表示されますが、キャラクタジェネレータ（略してCG）はあらかじめ固定されたパターンが格納されているROMCGと、ユーザーが自由に定義できるRAMCGがあります。

ROMCGの中には、英数字、カナ、セミグラフィックなどのパターンが入っており、標準状態でキーを入力するとROMCGのパターンが表示されます。RAMCGの内容は電源を切ると消えてしまいます。したがって、RAMCGを使用するにはあらかじめRAMCGにパターンを定義しておかなければなりません。この定義に使用するステートメントがDEFCHR\$です。また、RAMCGとROMCGとの表示を切り換えるステートメントがCGENです。

そこで例をあげて説明します。

```
CGEN 0
PRINT "A"
```

とします。

すると画面にはAという文字が表示されます。

次に

```
DEFCHR$(65)=STRING$(8,&HFF)+STRING$(8,&HF0)
+STRING$(8,&H81)
```

```
CGEN 1
```

```
PRINT "A" (CGEN 1以後はどのキーを押しても白ベタに表示されます。)
```

とすると画面には、縞模様のパターンが表示されます。

このパターンがRAMCGにDEFCHR\$で定義されたパターンです。

DEFCHR\$(65)の65はASCIIコードを示しています。ASCIIコード65はROMCGの"A"にあたります。

RAMCGは8×8ドットのパターンで256個まで定義できます。

またRAMCGとROMCGとの表示を切り換えるには **CTRL** キーを押しながらテンキーの **/** を押すことによっても可能です。

## 2.6 アンダーライン

テキスト画面で、行と行の間にアンダーラインを表示することができます。

アンダーラインは画面に表示する行が20行もしくは10行のときにのみ設定することができます。

つまり、アンダーラインを使用する場合、あらかじめ

```
WIDTH, 20 もしくは WIDTH, 10
```

としておきます。

これらのモードにすると、行と行の間にアンダーライン用のドットがあけて表示されます。

アンダーラインを表示する場合、KSENステートメントを使用します。



$KSEN \begin{cases} 0 & \text{または 省略} \\ 1 \end{cases}$
---



**KSEN 1** 

としますと、このステートメントを実行されたあとに入力された文字の下にアンダーラインが表示されます。

さらにアンダーラインの色は、KSEN命令の第2パラメータで指定できます。

**KSEN 1, 色** 

アンダーラインを表示しないモードにもどすには

**KSEN 0**  または **KSEN** 


とします。

### 3

## グラフィックと文字表示

LINE、SYMBOL、GET@、PUT@はグラフィックだけでなく文字表示にも使用できます。

たとえば

**LINE (0, 0) - (39, 9), "A", B** 

とすると(0, 0)と(39, 9)を対角線とするBOXを文字"A"でテキスト画面上に描きます。

また、


**SYMBOL (0, 0), "ABCD", 1, 1, 7, 0, "#"** 

とすると、"ABCD"という文字を"#"で描きます。

これらのステートメントの応用として、属性のみの変更に使えます。

たとえば、80文字のモードで、画面の上から3行目の文字にすべてアンダーラインを付ける場合


**KSEN 1, 7** 


**LINE (0, 2) - (79, 2), ""** 

とします。(このとき、WIDTHステートメントでアンダーラインの表示可能な画面モードにしておきます)つまり、文字列のかわりに""(ヌルコード)を指定すると、その時点の属性で、LINEやSYMBOLを描くことができます。

次に、GET@はテキスト画面のデータを配列変数に取り込むことができます。


たとえば、画面(0, 0) - (5, 5)の範囲に表示されている文字をそのまま、(25, 15) - (30, 20)の位置に書く場合

**DIM A (25)** 

**GET@(0, 0) - (5, 5), A** 

とし、配列変数Aに文字を読み込みます。

そして、

**PUT@(25, 15) - (30, 20), A** 

とします。

# 4章

## グラフィックス

# 4章

本機の特長の1つに、豊富なグラフィック処理機能があげられます。96Kバイトものグラフィック用メモリを標準装備していますので、複雑、多様な表現が可能です。本機のBASICでは、多彩なグラフィック処理コマンドによって、これらの機能をサポートしています。本章では、多くのプログラム例をまじえながら、これらコマンド群の使い方を説明します。本章で解説するコマンド・ステートメントを次に示します。

INIT、CLS、LINE、CIRCLE、POLY、PAINT、PALET、  
POSITION、PATTERN、SYMBOL、GET@、PUT@、PRW、COLOR

(注) 本章は、「2章、ディスプレイモード」の内容を土台にして書かれています。本文中で何かわからない点がありましたら、2章の記述を参照してください。

なお、40×20、80×20、40×10、80×10行の各モードではグラフィックを表示できません。

### 1

## 画面の初期化

まず最初に、画面を初期化するステートメントについて説明します。コマンド・ステートメントの中には、画面状態の設定も変えてしまうものもありますので、そのままでは思い通りのグラフィックが書けない場合もあります。そのような時、次のステートメントを実行することによって、画面状態はもとの設定にもどります。

INIT

また、**CTRL** + **D** とダイレクトに入力することによっても、INITステートメントと同様の結果が得られます。( **CTRL** + **D** は入門編1章「プログラミングを始めよう」参照)

上記INITステートメントは、画面の設定状態を初期化する命令ですので、画面に描かれている内容をクリアすることはできません。画面をクリアする場合は、次のステートメントを実行します。

CLS [n] (n=0~4)

[ ] で囲まれたパラメータは省略することができます。

CLSステートメントは、nの値によって処理が異なってきます。

CLS 0 .....グラフィック画面G1、G2、G3を同時にクリアします。

CLS 1 .....グラフィック画面G1のみをクリアします。

CLS 2 .....グラフィック画面G2のみをクリアします。



C L S 3.....グラフィック画面 G 3 のみをクリアします。

C L S 4.....グラフィック画面 G 1、G 2、G 3 およびテキスト画面を同時にクリアします。

C L S .....テキスト画面のみをクリアします。( **SHIFT** + **CLR HOME** と同じ処理をします)。

注) グラフィック画面 G 1、G 2、G 3 については、「2 章 2. グラフィック画面」の項を参照してください。

上記 I N I T、および C L S ステートメントによって、画面状態の初期化とクリアが行なわれます。したがって、これからいろいろなコマンド、ステートメントを実行していく中で、画面状態を元に戻したい場合は、上記ステートメントを実行してください。

## 2

## グラフィックステートメントの座標指定

グラフィック画面の座標系には、「2 章 8. 3 ユーザー座標系と画面座標系」で説明したように画面座標系とユーザー座標系の 2 種類あります。画面座標系は、画面上の 1 ドットが 1 座標に 1 対 1 に対応した絶対的な座標系ですので、不変ですが、ユーザー座標系は W I N D O W ステートメントの設定値によって変化します。これからこの章で述べていく各種グラフィックステートメントのグラフィック座標は、以上 2 つの座標系のどちらかに分類されますが、その分類の様子は次のようになります。

座標系	画面座標系	ユーザー座標系
グラフィック ステートメント	P O S I T I O N, S Y M B O L, G E T @, P U T @	P S E T, P R E S E T, L I N E, C I R C L E, P O L Y, P A I N T

このようにグラフィックステートメントによって座標系が異なりますので、W I N D O W ステートメントを使用した場合は、まず、そのステートメントがどちらの座標系に所属しているかを調べてから座標指定を行なうようにしてください。ただし、B A S I C 起動時には、ユーザー座標系は、画面座標系とまったく同じものですので、W I N D O W ステートメントでユーザー座標系を変えない限り、両者の区別はありません。座標系の初期化、すなわち、ユーザー座標系を画面座標系と同一にするには、I N I T ステートメントまたは **CTRL** + **D** を実行します。

(注) W I N D O W ステートメントの使用方法については「2 章 8. 3 ユーザー座標系と画面座標系」の項を参照してください。

## 3

## グラフィックステートメントの色指定

カラーモードにおいてグラフィック画面は、1 ドット単位に 8 色の色指定が可能です。これらの色指定には、カラーコードとパレットコードの 2 種類が用いられます。カラーコードは、色とコードが 1 対 1 に対応した絶対的なコードなので不変ですが、パレットコードは、色とコードの対応関係が、P A L E T ステートメントによって変化します。したがって、これからこの章で述べていく各種グラフィックステートメントの色指定がどちらのコードで行なわれているか、あらかじめ知っておく必要があります。次に、その区別の様子を示します。



色指定	カラーコード	パレットコード
グラフィック ステートメント	COLOR, CANVAS , KSEN	PSET, PRESET, LINE, CIRCLE, POLY, PAINT, SYMBOL, PRW

ただし、BASIC起動時には、パレットコードはカラーコードとまったく同じ色になっていますので、PALETステートメントでパレットコードの色指定を変えない限り、両者の区別はありません。パレットコードを初期化するためには、INITステートメントまたはPALETステートメントまたは **CTRL** + **D** を実行します。

また、グラフィックステートメントにおいて、色指定（パレットコード）を省略した場合は、自動的に、COLORステートメントの第1パラメータで指定した値（カラーコード）を、そのグラフィックステートメントのパレットコードとみなして実行します。

COLORステートメントの設定は、次の通りです。

**COLOR** [c1] [, c2]

c1：テキスト文字の表示色（カラーコード）

c2：画面の背景色（カラーコード）

これより、グラフィックステートメント中の色指定を省略すると、パレットコードがカラーコードと等しい場合テキスト画面の文字の色と同じ色でグラフィックが描かれることになります。

たとえば、COLORステートメントの第1パラメータをカラーコード1に設定した場合、パレットコードが初期状態ならば、パレットコードの指定を省略したグラフィックステートメントは青色として実行されます。また、PALETステートメントによって、パレットコードの1がカラーコードの4に設定されていれば、パレットコードの指定を省略した、グラフィックステートメントは緑色として実行されます。なお、COLORステートメントは、BASIC起動時には、次のように設定されています。

**COLOR 7, 0**

また、COLORステートメントのかわりに、以下のようなダイレクト実行によっても同様な結果が得られます。

<b>CTRL</b> + テンキー <b>0</b>	(=COLOR 0)
<b>CTRL</b> + テンキー <b>1</b>	(=COLOR 1)
⋮	⋮
<b>CTRL</b> + テンキー <b>7</b>	(=COLOR 7)

(注) PALETステートメントの設定方法については、後述「8. 色を瞬時に変える」の項を参照してください。

(注) 本章では、特に断わらない限り、カラーモードにおけるグラフィック表示について述べます。  
(カラーモードについては、「2章 6. カラーモード」の項を参照してください。)

## LINEステートメント

LINEステートメントは、ただ単に実線を引くだけではなく、パラメータの設定によって次の動作を行なうことができます。

- ①ラインスタイルを設定することによって、点線、破線、一点鎖線など自由な形のラインを引くことができます。
- ②任意のラインスタイルで、長方形を描くことができます。
- ③長方形を描いて、その中を任意のパレットコード、中間色コードまたはタイリングパターンで塗りつぶすことができます。
- ④座標ばかりを連続入力することによって、任意の折れ線や多角形を描くことができます。

## 4.1 線を描く

線を描く場合、LINEステートメントのパラメータは次のようになります。

**LINE [(x<sub>1</sub>, y<sub>1</sub>)] - (x<sub>2</sub>, y<sub>2</sub>) [, mode, c, ls]**

x<sub>1</sub>, y<sub>1</sub>: 始点座標 (ユーザー座標系)

x<sub>2</sub>, y<sub>2</sub>: 終点座標 (ユーザー座標系)

mode: PSET, PRESET, XOR

c: パレットコード      ls: ラインスタイル

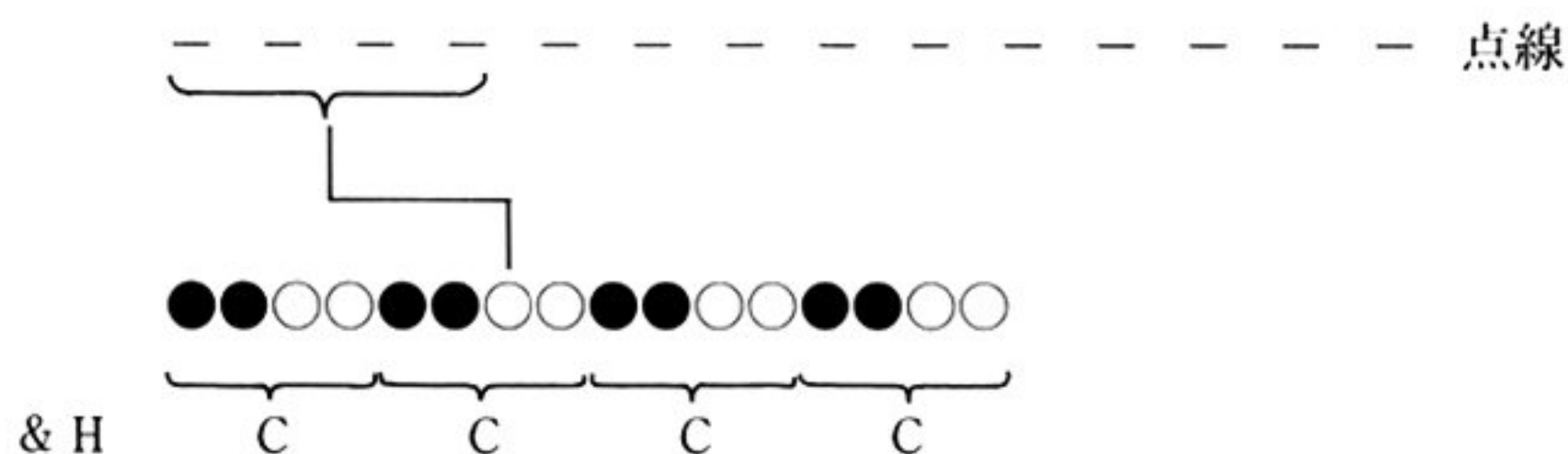
ただ単に実線を引くだけならば、入門編「2章. 絵を描く、色を塗る」で説明したようにたとえば次の命令を実行するだけでOKです。

**LINE (0, 0) - (319, 199), PSET**

これで、2点 (0, 0)、(319, 199) を結ぶ実線が引かれます。線の色は、パレットコードを省略しているため、テキスト画面の文字の色に対応したパレットコードが与えられます。(初期状態では白色です) また、始点座標が省略されると、直前に実行したLINEステートメントの終点座標が新たに始点座標としてみなされます。

第3パラメータ (mode) には、PSET, PRESET, XORの3通りの指定が可能です。PSET, PRESETについては「点を描く」で説明した通りです。XORを指定すると、他のグラフィック図形と重なった直線部分の色が反転します。たとえば、すでに白 (青+赤+緑) でペイントされている所へ、XORを使って青のラインを引くと、重なった部分のラインの色は黄 (赤+緑) になります。このパラメータが省略されると、直前に設定したmodeの影響をうけます。

第5パラメータ (ls) には、ラインスタイルを指定します。ラインスタイルとは、点線、破線などを16ビットのビットパターンで表わしたものです。たとえば、次のような点線のラインスタイルは&HCCCCで表わされます。








このように、ラインスタイルは、16ドットの範囲内で任意のくり返しパターンを設定することができます。このパラメータを省略すると、実線（ラインスタイル=&HFFF F）で引かれます。以下に主なラインスタイルとその破線のパターンを示します。

ラインスタイル	破線のパターン
\$HFFF F	_____ 実線
\$HFF 00	— — — — —
\$HCCCC	----- 点線
\$HAAAA	-----
\$HFFCC	----- 一点鎖線
\$HE4E4	-----
\$HFCCC	----- 二点鎖線
\$HFF24	-----


では、実際に直線を引いてみましょう。まず、画面を初期化して、グラフィック画面をクリアします。

```
INIT 
CLS0 
```

次に、2点(0, 0)、(190, 190)を結ぶ1点鎖線を黄色で引きます。

```
LINE (0, 0) - (190, 190), PSET, 6, &HFFCC 
```

どうですか？うまく引かれましたか？もし、思いどおりの直線がひかれなければ **CTRL** + **D** を押してみてください。さて、正しく表示されたら、今度は同じ座標間をXORモードを使って黄色の実線で結びます。

```
LINE (0, 0) - (190, 190), XOR, 6 
```

どうになりましたか？今まで1点鎖線が表示されていたところが透明になり、今まで透明だった部分に黄色が表示されましたね。このように、XORを指定すると、重なった部分のXOR論理がとられ、色が反転してしまいます。

### サンプル プログラム

各種ラインスタイルを実際に表示させてみます。

```
10 'LINE STYLE
20 WIDTH 40,25,0:CLS0:INIT
25 PRINT"ラインスタイル";SPACE$(10);"パターン"
30 RESTORE 100
40 FOR I=2 TO 9
50 LOCATE 0,I*2
60 READ A$
70 PRINT A$
80 LINE(60,I*16+4)-(319,I*16+4),PSET,7,VAL(A$)
90 NEXT
100 DATA &HFFF,&HFF00,&HCCC,&HAAA,&HFFCC,&HE4E4,&HFCCC,&HFF24
```



## 4.2 長方形を描く

長方形を描く場合、LINEステートメントのパラメータは次のようになります。

```
LINE [(x1, y1)] - (x2, y2), mode [, c], B [, ls]
```

B：長方形を描く指定

指定した2点間の対角線を引くか、長方形を描くかは、記号"B" (BOXの略) があるかないかによって決まります。他のパラメータは、線を引く場合とまったく同じです。


では、実際に長方形を描いてみましょう。まず、画面の初期化とクリアを行ないます。

```
INIT   
CLS 0 
```

次に、2点(0, 0) (190, 190)を対角とする長方形をシアンで描きます。

```
LINE (0, 0) - (190, 190), PSET, 5, B 
```

さらに、その上から、赤の点線でなぞってみます。

```
LINE (0, 0) - (190, 190), PSET, 2, B, &HCCCC 
```

これで、赤とシアンの点線で長方形が描かれました。

## 4.3 長方形を塗りつぶす

長方形を塗りつぶす場合、LINEステートメントのパラメータは次のようになります。

```
LINE [(x1, y1)] - (x2, y2), mode [, c], BF
```

c：パレットコードまたは中間色コード

BF：長方形を塗りつぶす指定

長方形を描いてその中を塗りつぶすには、記号"BF" (Box Fillの略) を指定します。ただし、この場合、指定できる色はパレットコードだけでなく、中間色コードも指定できます。

たとえば、2点(0, 0)、(190, 190)を対角とする長方形をシアンと赤の中間色コードで塗りつぶす場合は、次のステートメントを実行します。

```
LINE (0, 0) - (190, 190), PSET, &H25, BF 
```

ただし、BFを指定する場合は、ラインスタイルを指定することはできません。そのかわり、タイリングパターンを設定することができます。タイリングパターンというのは、より複雑な中間色や色模様を出すための文字列データで、最大8×8ドットの色パターンを定義することができます。最小のタイリングパターンは横8ドットの色パターンで、これを指定するには青、赤、緑各1バイトずつの計3バイトの文字列データが必要です。したがって、8×8のタイリングパターンを定義する場合は、24バイトの文字列データが必要です。このように、中間色が2色のドットを交互に配置するだけなのに対して、タイリングパターンは、横8ドットの色パターンを自由に設定することができます。タイリングパターンを設定すると、その色パターンを使って長方形を描き、かつ塗りつぶします。タイリングパターンを使用する時、LINEステートメントは次のようになります。

```
LINE [(x1, y1)] - (x2, y2), mode, BF, t$
```

t\$：タイリングパターン (文字列)

タイリングパターンの文字列は、CHR\$やHEXCHR\$関数によって用意するのが一般的です。

たとえば、横8ドットのパターンを、1ドット1色ずつにして8色をすべて使って定義する場合は次のようにします。

横8ドットのパターン	黒	青	赤	マゼンタ	緑	シアン	黄	白	
	↓	↓	↓	↓	↓	↓	↓	↓	
青データ	0	1	0	1	0	1	0	1	= &H55
赤データ	0	0	1	1	0	0	1	1	= &H33
緑データ	0	0	0	0	1	1	1	1	= &H0F

タイリングパターンt\$は次のように表わされます。

t\$=HEXCHR\$("55330F")

そのタイリングパターンで、2点(0, 0)、(190, 190)を対角とする長方形を塗りつぶすには、次のステートメントを実行します。

LINE(0, 0) - (190, 190), PSET, BF, HEXCHR\$("55330F")

## 4.4 折れ線を描く

LINEステートメントは、グラフ等に必要な折れ線を簡単に描くことができます。その場合のパラメータは次のようになります。

LINE [(x<sub>1</sub>, y<sub>1</sub>)] - (x<sub>2</sub>, y<sub>2</sub>), (x, y)..... (x<sub>n</sub>, y<sub>n</sub>)

x<sub>1</sub>, y<sub>1</sub>: 折れ線の始点座標

x<sub>i</sub>, y<sub>i</sub>: 折れ線の頂点座標 (i=2, 3.....n-1)

x<sub>n</sub>, y<sub>n</sub>: 折れ線の終点座標

このように、座標を連続して入力すると、各座標間を結んだ折れ線を描きます。これを利用すれば、任意の形の多角形を簡単に描くこともできます。たとえば、4点(200, 0), (100, 100), (0, 100), (100, 0)を頂点とする平行四辺形を描くには、次のように入力します。

LINE(200, 0) - (100, 100) - (0, 100) - (100, 0) - (200, 0)

# 5

## 正多角形を描く

### POLYステートメント

正多角形に限り、POLYステートメントを使って簡単に描くことができます。

POLY (x, y), r[, c, Δθ, θs, θe]

x, y: 正多角形の中心座標 (ユーザー座標系)

r: 中心から頂点までの距離

c: 正多角形の辺の色 (パレットコード)



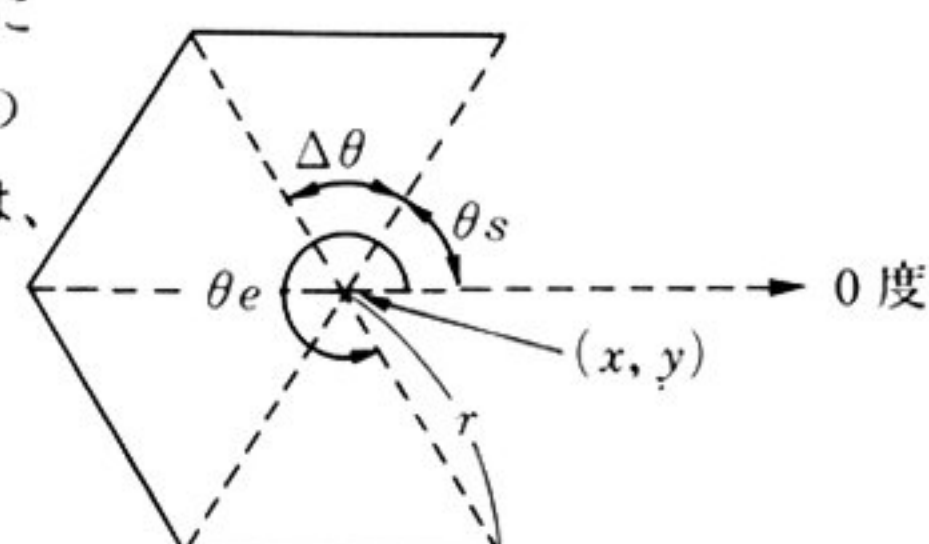
$\Delta \theta$  : ステップ角 (0 ~ 360 度)

$\theta_s$  : 初期角 (0 ~ 360 度)

$\theta_e$  : 終了角 (0 ~ 360 度)

多角形は、前述の LINE ステートメントでも描くことができますが、LINE の場合は各頂点の座標を計算して入力しなければなりません。しかし、POLY ステートメントでは、円を描くと同じ要領で入力することができますので、正多角形の中心 (x, y) と半径 r 等の設定だけで簡単に描くことができます。(円については「6. 円を描く」で述べます)。各パラメータについて以下に示します。

ステップ角 ( $\Delta \theta$ ) は、正多角形の頂点間の角度で、0 度に近いほど真円に近づきます (ただし、0 度を指定するとただの線分になってしまいます)。初期値 ( $\theta_s$ ) と終了角 ( $\theta_e$ ) は、x 座標の増分方向 (画面の右方向) を 0 度とした場合の、多角形を描き始める角度と、描き終える角度を示します。



したがって、一般的に、正 n 角形を描くには、次のように指定します。

$$\Delta \theta = 360 / n$$

$$\theta_e = \theta_s + 360$$

では、実際に、正多角形を描いてみます。まず、画面を初期化してクリアしてください。

INIT

CLS

今度は、右側半分だけの半円を描いてみます。右側だけの半円ですから、初期角は 90 度、終了角は 270 度で指定します。

**POLY (200, 100), 80, 6, 1, 90, 270**

これで黄色の半円が描かれました。

(注) POLY および CIRCLE ステートメントにおける頂点距離または半径は、画面モードによって、実際の座標と対応していない場合があります。詳しくは、『BASIC リファレンスマニュアル』の 2.8.7 CIRCLE ステートメントの項を参照してください。



サンプル  
プログラム

P O L Yで多角形を重ねます。

```

10 INIT:CLS4:WIDTH 40,25,0,0
20 RESTORE 160
30 FOR KAI=1 TO 5
40 READ STE1,STA1,OWA1,STE2,STA2,OWA2
50 C=INT(RND*6)+1
60 FOR X=1 TO 14
70 FOR Y=1 TO 9
80 POLY(X*20,Y*20),20,C,STE1,STA1,OWA1
90 POLY(X*20+2,Y*20),20,C+1,STE2,STA2,OWA2
100 NEXT Y
110 NEXT X
120 PAUSE 20
130 CLS4
140 NEXT KAI
150 GOTO 20
160 DATA 120,0,360,120,90,450
170 DATA 90,0,360,90,45,405
180 DATA 72,0,360,72,90,450
190 DATA 60,0,360,60,30,390
200 DATA 144,0,720,144,90,810

```

サンプル  
プログラム

時計を描きます。

```

10 INIT:CLS4:WIDTH 40,25,0,0
20 POLY(160,100),80,7,30
30 T$=TIME$
40 IF T$=T1$ THEN 30
50 X=VAL(LEFT$(T$,2))
60 Y=VAL(MID$(T$,4,2))
70 Z=VAL(RIGHT$(T$,2))
80 POLY(160,100),40,0,0,90-X*30-INT(Y/2)
90 POLY(160,100),50,0,0,90-Y*6
100 POLY(160,100),60,0,0,90-Z*6
110 POLY(160,100),40,5,0,90-X*30-INT(Y/2)
120 POLY(160,100),50,4,0,90-Y*6
130 POLY(160,100),60,6,0,90-Z*6
140 LOCATE 16,16:PRINT T$
150 X1=X:Y1=Y:Z1=Z:T1$=T$
160 GOTO 30

```

## CIRCLEステートメント

円、楕円および円弧を描くステートメントは、CIRCLEステートメントです。

CIRCLE [@] (x, y), r [, c, f,  $\theta$ s,  $\theta$ e]

f: 偏平率 (=たて径/横径)

このように、CIRCLEステートメントのパラメータは、POLYステートメントと似ています。

違いは、POLYステートメントのステップ角が、CIRCLEステートメントでは偏平率に変わっている点だけです。また、CIRCLEステートメントでは、@記号がつくつかないかで、半径及び偏平率の扱いが異なってきます。

## (1) @記号がつかない場合

画面に表示される円の大きさは、320×200ドットモードの時を基準にしています。したがって、半径が同じならば、どの画面モードで描いても、画面に表示される円の大きさは同じになります。このことは画面上の座標と半径の大きさとが、必ずしも一致しないことを示しています。偏平率を使用した場合も同様です。偏平率を省略すると、1が設定されます。

## (2) @記号がつく場合

画面に表示される円の大きさは、各画面モードの座標に従います。したがって、同じ半径でも画面モードによって円の大きさは異なってきます。偏平率を指定した場合でも同様です。したがって、偏平率を1に設定したからといって、画面モードによっては真円にならない場合があります。偏平率を省略すると、画面上に真円が描かれるように、自動的に偏平率を設定します。

なお、詳しくは、『BASICリファレンスマニュアル』の2.8.7 CIRCLEステートメントおよび2.8.8 CIRCLE@ステートメントの項を参照してください。

サンプル  
プログラム

専用ディスプレイテレビにおいて、CIRCLEステートメントとCIRCLE@ステートメントにおける半径の取扱いを示すプログラムです。

```
10 'CIRCLE
20 INIT:CLS4:OPTIONSCREEN0
30 G=0:RES=1:GOSUB 70
40 G=0:RES=2:GOSUB 70
50 G=1:RES=2:GOSUB 70
60 END
70 'SUB
80 FOR X=40 TO 80 STEP 40
90 WIDTH X,25,G,RES
100 PRINT "WIDTH ";X,"25,";G,"";RES
110 PRINT "  RED....CIRCLE (200,100),100,2"
120 PRINT "  GREEN...CIRCLE@(200,100),100,4"
130 CIRCLE(200,100),100,2
140 CIRCLE@(200,100),100,4
150 A$=INKEY$:IF A$=""THEN 150
160 NEXT
170 RETURN
```



## P A I N Tステートメント

グラフィック曲線で囲まれた任意の形の図形を塗るには、P A I N Tステートメントを使います。  
ここでは中間色を指定します。

本機では、赤、青、緑の3色を1ドット単位に自由な組合わせで基本の8色以外の中間色を作ることができます。これをタイリングペイントといいます。

- [1] P A I N T (x, y), c, b [b<sub>1</sub>, b<sub>2</sub>, …, b<sub>6</sub>]  
 [2] P A I N T (x, y), t\$, b [, b<sub>1</sub>, b<sub>2</sub>, …, b<sub>6</sub>]

c:パレットコードまたは中間色コード

t\$ ……タイリングパターン

b, b<sub>1</sub>, b<sub>2</sub>, …, b<sub>6</sub>:境界色 (パレットコード)

[1] の書式で、中間色を指定するとき c を次のように表わします。

&H m n (または m \* 16 + n)

m = 0 ~ 7 のパレットコード

n = 0 ~ F のコード

こうすれば、m と n の2色を混合した中間色を出すことができます。

(中間色一覧表は「2章・5.中間色コード」を参照してください。)

赤色を描いてその中を中間色 (紫) で塗ります。

```
10 WIDTH 40,25,0:INIT
20 CLS4
30 CIRCLE(200,100),80,2
40 PAINT(200,100),&H12,2
```

[2] の書式で、中間色を指定するときは、タイリングパターン t\$ は CHR\$ や HEXCHR\$ 関数によって表わします。

たとえば、黄緑色は黄色と緑色を1ドット単位で交互に点灯することにより表示できます。

CHR\$ (&H 0 0) + CHR\$ (&H A A) + CHR\$ (&H F F) …黄緑色

または HEXCHR\$ (" 0 0 A A F F ")

青→0 0 0 0 0 0 0 0 →&H 0 0

赤→1 0 1 0 1 0 1 0 →&H A A

青→1 1 1 1 1 1 1 1 →&H F F

黄 緑 黄 緑 黄 緑 黄 緑

このデータを P A I N T 文に入れますと

P A I N T (x, y), CHR\$ (&H 0 0) + CHR\$ (&H A A) + CHR\$ (&H F F)

または

P A I N T (x, y), HEXCHR\$ (" 0 0 A A F F ")



## P A L E Tステートメント

P A L E Tステートメントは、グラフィック画面上の色を瞬時に他の色に変換します。

P A L E T p, c, b

p: パレットコード (0 ~ 7)

c: カラーコード (0 ~ 8)

b: ボーダーカラー (0 または 1)

P A L E Tステートメントは、色の変換を行ないますが、実際に塗りつぶして変換するわけではありません。したがって、瞬時の色変換が可能です。変換は、パレットコードをどのカラーコードに対応させるか、で行ないます。パレットコードおよびカラーコードについては「2章 ディスプレイモード」の項を参照してください。

たとえば、パレットコード1は、初期状態ではカラーコード1すなわち青色に対応していますが、赤色（カラーコード2）に変換するには、次の命令を実行します。

P A L E T 1, 2 

また、P A L E Tステートメントによりグラフィックの透明と青の2色を、黒色に変換することができます。黒色は、カラーコード8に割り当てられており、P A L E T 0, 8 または P A L E T 1, 8 のどちらかを指定します。前者が透明を黒に、後者が青を黒に変換します。ただし、黒色といっても、コンピュータモードでは透明とまったく変わりません。透明との違いができるのはスーパーインポーズ時のみです。透明はバックのテレビ画面を透過しますが、黒色は透過しません。


また、第3パラメータが1の時、画面のボーダー部分を黒色にします。0の時は透明のままです。これによって、スーパーインポーズ時にテレビ画面を黒ぬきにすることができます。

また、P A L E Tステートメントは次のように設定することもできます。

P A L E T @ c1, c2, c3, c4, c5, c6, c7, c8

c1, c2, ..... c8: カラーコード

この指定方法は、複数のパレットコードを一度に変換する場合に用いられます。パレットコードは、パラメータの位置に対応しています。すなわち、第1パラメータがパレットコード0に、第2パラメータがパレットコードの1に、第8パラメータがパレットコード7にそれぞれ対応しています。たとえば、パレットコード1を赤に、パレットコード7を緑に変換するためには、次の命令を実行します。

P A L E T @ 0, 2, 2, 3, 4, 5, 6, 4 

花火の点滅する感じをお楽しみください。

```

10 '花火
20 WIDTH 40,25,0:CLS4:INIT
30 SCREEN1,1:KLIST0:PRINT "Wait a moment!"
40 DEFFNX(CX)=COS(RAD(CX))*CR
50 DEFFNY(CY)=SIN(RAD(CY))*CR
60 CCX=INT(RND*300)+10
70 CCY=INT(RND*150)+10
80 H=INT(RND*10)+6
90 SCREEN 1,0:KLIST0
100 FOR CR=1 TO H*7 STEP H
110 FOR DEG=0 TO 360 STEP H
120 C=INT(RND*7)+1
130 PSET(FNX(DEG)+CCX,FNY(DEG)+CCY,C)
140 NEXT DEG
150 NEXT CR
160 SCREEN 1,1:CLS4
170 SOUND 7,8HFE:PLAY"-E1"
180 FOR UP=200 TO CCY STEP -1
190 C=INT(RND*7)+1
200 PSET(CCX,UP,C):PSET(CCX+1,UP,C)
210 PRESET(CCX,UP,C):PRESET(CCX+1,UP,C)
220 NEXT UP
230 SCREEN0,0
240 SOUND 7,8HF7:SOUND 6,43:SOUND 8,16:SOUND 11,0:SOUND 12,150:SOUND 13,0
250 FOR I=1 TO 15
260 FOR J=1 TO 7
270 PALET J,((J+1)MOD 7)+1
280 NEXT J
290 NEXT I
300 FOR I=1 TO 7
310 PALET I,0
320 PAUSE 1
330 NEXT I

```

## 9

## テキスト文字とグラフィック画面の優先順位を決める

### PRWステートメント

PRWステートメントは、テキスト画面上の文字とグラフィック画面上の図形が重なった時、どちらを優先して表示させるかを定めるステートメントです。

**PRW** [n]

n=0~255

グラフィック画面の各色とテキスト文字との優先順位は、nの値で決定されます。nの値はコンピュータ内部では8ビットで表現されており、各ビットは次のように各パレットコードに対応しています。



	7	6	5	4	3	2	1	0	
n:	白	黄	シアン	緑	マゼンタ	赤	青	透明	(初期状態のパレットコード)

最下位ビットが透明に対応し、最上位ビットが白に対応しています。この時、あるビットの内容が1に設定されると、その色をもつグラフィックが優先され、0に設定されるとテキスト文字が優先されて表示されます。たとえば、グラフィックの青と緑をテキストより優先させて表示させるには、次のように設定します。

**PRW &B00010010** (=**PRW 18**)

PRWステートメントは、パレットコードに対する命令ですので、たとえば、PALET 1, 2を実行してパレットコード1を赤に変換すると、その部分の赤がテキストより優先されます。ただし、パレットコード2の赤はテキストに隠れたままです。このように、PRWステートメントを利用すると、見かけ上同じ色でもテキストとの優先順位を変えることができます。

優先順位を初期状態にもどすには、PRWを実行します。

## 10 文字パターンの描画

### SYMBOLステートメント

SYMBOLステートメントは、グラフィック画面上に文字列を指定の角度とサイズで描きます。

$$\text{SYMBOL (x, y), x\$, h, v, } \left\{ \begin{array}{c} c' \\ t\$ \end{array} \right\}, \theta, \text{mode}$$

x, y : 文字列の表示を開始する左上のドットの座標 (画面座標系)

x\$ : 表示する文字列

h : 横方向の倍率 (1, 2, 3……)

v : 縦方向の倍率 (1, 2, 3……)

c' : 文字の色 (パレットコードまたは中間色コード)

$\theta$  : 描く方向の指定 (0 ~ 3)

mode : PSET, PRESET, XOR, 文字式または " "

t\$ : タイリングパターン

h, vは、それぞれ横、縦方向の倍率を指定します。設定値はすべて整数とみなされます。小数は四捨五入されます。この値によっては文字が画面から、はみ出してしまうので注意してください。

t\$はタイリングパターンで、HEXCHR\$などで指定します。タイリングパターンについては「7.色を塗る」の項を参照してください。

$\theta$ は、表示する文字の方向を示しています。

$\theta = 0$ では、通常表示

$\theta = 1$ では、90度左に回転したもの


$\theta = 2$ では、180度左へ回転したもの

$\theta = 3$ では、270度左へ回転したもの


となります。



mode は、文字列の表示モードを示し、PSET, PRESET, XOR を指定するとグラフィック画面に表示し、文字式または " " ヌルコードを指定するとテキスト画面に表示します。次に例を示します。

**SYMBOL (100, 100), "A", 1, 1, 1, 0, PSET** 

これはグラフィック画面に、青色のAという文字をたて横1倍で表示します。

**SYMBOL (100, 100), "B", 2, 2, 4, 1, PSET** 

これら、グラフィック画面に緑色のBという文字をたて横2倍の大きさで表示します。その時Bは90度左に回転されて表示します。

**SYMBOL (0, 0), "A", 1, 1, 1, 0, "B"**

これは、テキスト画面にキャラクタBでAという文字を表示します。

## 1 1

## GET@とPUT@

GET@はグラフィック画面の情報を配列変数に読み込むステートメントです。

また、PUT@というステートメントはGET@の逆で、配列変数の情報を画面に描くステートメントです。

通常、これらの2つのステートメントを組み合わせて用いることによって、グラフィック画面のデータを別の位置に移動させることができます。

これらのステートメントは、画面データを一度配列変数に入れるので、データが入るだけの配列変数領域をあらかじめ定義しておく必要があります。

たとえば、次の例では、画面左上に描いた円を赤でぬり、それと同じ円を別の場所に描いています。

```
10 INIT:CLS
20 DIM A%(100)
30 CIRCLE(10,10),5:PAINT(10,10),2,7
40 GET@(0,0)-(20,20),A%,7
50 FOR I=0 TO 100 STEP 20
60 PUT@(I,I)-(I+20,I+20),A%,PSET,7
70 NEXT
```

GET@およびPUT@の書式は

GET@(x <sub>1</sub> , y <sub>1</sub> ) - (x <sub>2</sub> , y <sub>2</sub> ), 配列名, パレットコード
PUT@(x <sub>1</sub> , y <sub>1</sub> ) - (x <sub>2</sub> , y <sub>2</sub> ), 配列名, <div style="display: inline-block; vertical-align: middle;">PSET, パレットコード PRESET XOR OR AND NOT</div>

x<sub>1</sub>, y<sub>1</sub>: 読みこむ領域の左上隅の座標 (画面座標系)

x<sub>2</sub>, y<sub>2</sub>: 読みこむ領域の右下隅の座標 (画面座標系)

PUT@ステートメントでは表示のときにPSET, PRESET, XOR, OR, AND, NOTとモードがあり、画面にすでにあるドットと配列中のドットの演算結果を画面に表示することもできます。また、パレットコードを省略するとテキスト画面に対するステートメントになります。

## POSITION・PATTERNステートメント

まず、グラフィック画面にドットパターンを描く際には描画開始位置を指定します。  
それにはPOSITIONステートメントを用意します。

**POSITION x, y**

x, y: ドットパターン表示の左上隅の座標 (画面座標系)

POSITIONステートメントの実行により描画開始位置を指定した後、PATTERNステートメントでドットパターンを描画します。

**PATTERN n, x\$ [, y\$.....]**

n: ドットパターンのたて方向の段数を指定し、 $n < 0$  のとき下の方向に重なり  $n > 0$  のとき上の方向に重なります。

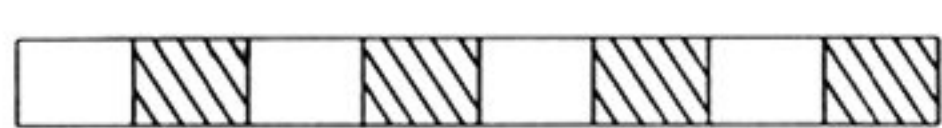
x\$, y\$: ドットパターンを表わす文字式。


たとえば

**POSITION 10, 10**

**PATTERN -1, CHR\$ (&H55)**

と入力すると x 座標 10, y 座標 10 を描画開始位置として、ドットパターン



() がセット。&H55 = &B01010101) が表示されます。

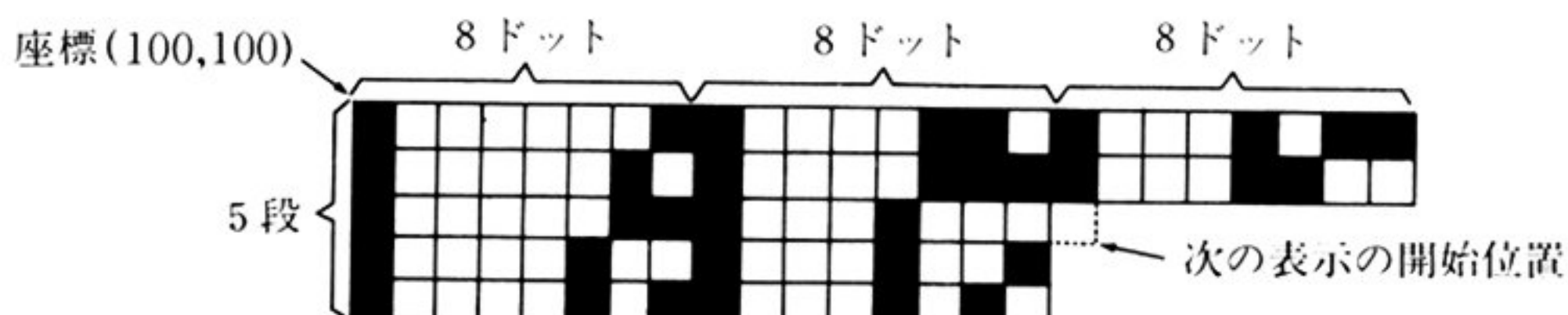
また、ドットパターンの積み重ねは段数を指定する n の値より、積み重ねの方向と段数が決められます。n が負のとき上から下へ、n が正のとき下から上へ積み重ねられます。

表示色は、テキスト画面の文字の色に対応したパレットコードで描れます。

たとえば

```
10 WIDTH 40: INIT
100 A$=HEXCHR$("8182838485868788898A8B8C")
110 POSITION 100,100
120 PATTERN-5,A$
```

を実行すると下の様に表示されます。





# 5章

## 日本語処理

入門編「漢字を表示する」では漢字BASICを利用して、漢字、ひらがな等を表示してみました。この漢字BASIC（CZ-8FB02）は、漢字、ひらがな等の文字をBASICプログラム上で容易に扱えるように、カナ-漢字変換機能や外字登録機能を持っています。この章では、これら全角文字と呼ばれる文字の扱いについて説明します。

### 1

### 全角文字とは

**全角文字**とは、漢字やひらがなのように、そのフォントが16×16ドットで構成された文字のことを指します。それに対して、そのフォントが8×8または16×8ドットで構成された文字を、**半角文字**と呼んでいます。

従来のコンピュータでは、通常、英数字・カタカナ・記号・セミグラフィック等の半角文字のみ扱っており、漢字・ひらがな等の全角文字はプログラム中ではコード（JIS漢字コード、区点コード等）として扱うことしかできませんでした。（たとえば、「亜」という漢字はKANJ1\$（1601）で表わします）しかし、本機の漢字BASICでは、これらの全角文字をプログラム中で半角文字と同様に文字の形で扱うことができます。それは、次のような場合において可能です。

- (1) PRINT文等のダブルクォーテーション（"）で囲まれた部分
- (2) ファイル名及びディレクトリ名
- (3) REM文
- (4) DATA文
- (5) 文字変数

このように、本機の漢字BASICでは、プログラム中に漢字やひらがなを混入させることができますが、BASIC内部ではこれらの全角文字は2バイトのコードとして処理されています。この意味から、従来の文字（半角文字）が1バイトコード文字と呼ばれているのに対して、全角文字は2バイトコード文字とも呼ばれています。

したがって全角文字の1文字分は半角文字の2文字分とみなされますので、たとえば、ファイル名の長さ等、文字数に注意する必要があります。また、本機の漢字BASICでは、全角文字を扱う方式として、従来の半角文字のセミグラフィック部分の1バイトコードを利用するSHIFT JISコード体系をとっています。したがって、全角文字を扱う場合はセミグラフィック文字を混在させることができません。（ただし、英数字、カタカナ、記号等は使用できます）全角文字を表示させるか否かは、次のBASICコマンドによって決められます。

**KMODE 0** ……半角文字のみ扱うモード（セミグラフィック文字使用可）

**KMODE 1** ……全角文字も扱うことができるモード（セミグラフィック文字使用不可）

また、全角文字の場合その文字のフォントがたて16ドットで構成されていますので、表示画面が標準ディスプレイモードの時には、全角文字は次の画面モードの時のみ表示することができます。



```

WIDTH 80, 12, 0, | 1
                      | 0
WIDTH 40, 12, 0, | 1
                      | 0
WIDTH 80, 10, 0, | 1
                      | 0
WIDTH 40, 10, 0, | 1
                      | 0

```

## 2

## 全角文字の入力方式

### (1) ひらがな、カタカナ、英数字等の全角文字の入力方式

次の3通りの方式があります。また、これらの方式では半角文字の入力も可能です。

- ①英数字直接入力方式—アルファベット、数字等の入力を簡単に行なえる。
- ②ローマ字—カナ変換方式—ローマ字表記にて入力し、それをひらがなまたはカタカナに変換する。
- ③カナ入力方式—カナ キーをロックしてカナ表記にて入力し、それをひらがなまたはカタカナに変換する。

### (2) 漢字への変換機能

次の2通りの方式があります。

- ①コード入力方式—漢字のコード（JIS漢字コードまたは区点コード）を入力して、直接、漢字、ひらがな等の全角文字に変換する。
- ②カナ—漢字変換方式—(1)の②ローマ字—カナ変換方式および③カナ入力方式において入力した、ひらがなまたはカタカナを漢字に変換する。  
この方式には次の方式があります。

- ・一字変換方式…ひらがなまたはカタカナの文字列の先頭の1文字のみを判断して、その読みを持つ漢字をJIS漢字コードの配列通りに羅列して、変換する。
- ・音訓変換方式…漢字の音読み、訓読みを入力し変換する。

なお、一字変換方式は、コンピュータの内部プログラムとして存在しますのでディスク等は一切不要ですが、音訓変換方式は、システムディスクの中の音訓辞書が必要です。

### (3) 日本語入力モードに入るための条件

〔条件1〕

標準ディスプレイモードの場合、画面モードを次のいずれかに設定すること。（高解像度をディスプレイモードの場合は、制限ありません）

```

○WIDTH 80, 12, 0, | 1
                      | 0
○WIDTH 40, 12, 0, | 1
                      | 0
○WIDTH 80, 10, 0, | 1
                      | 0
○WIDTH 40, 10, 0, | 1
                      | 0

```

### 〔条件2〕

CONSOLEステートメントによって、表示画面の最下行が変換フィールドエリアとして確保されていること。

(例)

- 25行モードならば、CONSOLE 0, 24
- 12行モードならば、CONSOLE 0, 11
- 20行モードならば、CONSOLE 0, 19
- 10行モードならば、CONSOLE 0, 9

ただし、CONSOLEステートメントだけでなく、INIT命令や **CTRL** + **D** キー入力によっても、変換フィールドエリアは確保されます。

### 〔条件3〕

"KMODE 1" を実行してあること。

以上の条件が満たされているならば、

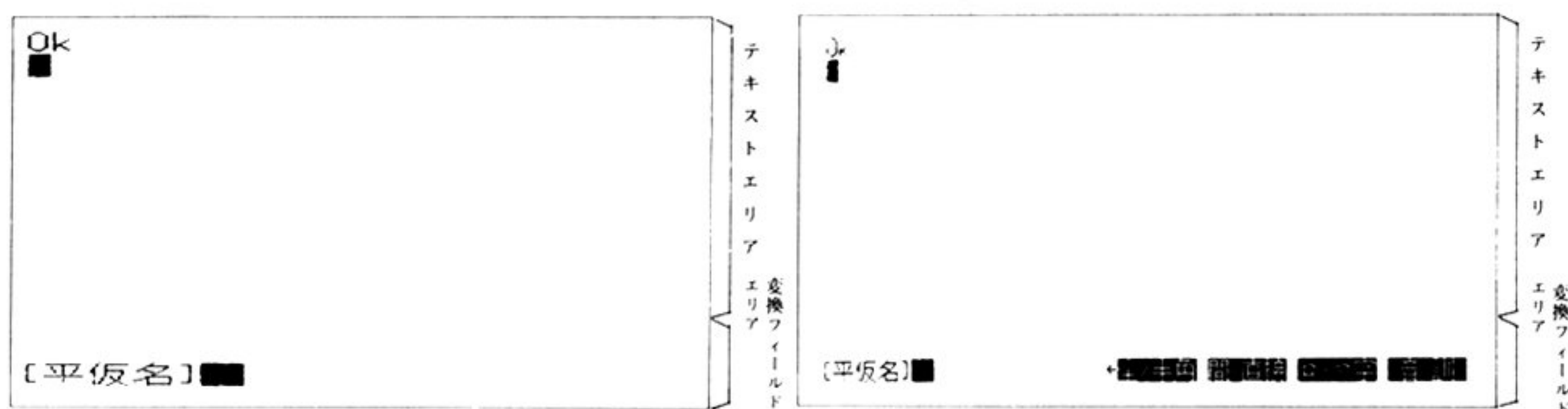
- (a) **CTRL** キーを押しながら **XFER** キーを押す。 (**CTRL** + **XFER**)  
または
- (b) **SHIFT** キーを押しながら **XFER** キーを押す。 (**SHIFT** + **XFER**)

と、日本語入力モードに、はいることができます。

なお、BASICの起動時では、上記の条件〔1〕～〔3〕までを満たしています。

日本語入力モードに入ると、表示画面の最下行に変換フィールドが現われます。

全角文字への変換は、すべてこの変換フィールド内で行なわれます。また、日本語入力モードから通常の状態に戻るには、再び前記(a)または(b)を実行します。



(a)40桁モードの場合の入力画面

(b)80桁モードの場合の入力画面

日本語入力モードにおける表示画面例

〔注〕 日本語入力モードにおける表示画面は、40桁と80桁モードとでは若干異なりますが、動作には変わりはありません。

## 3

## 入力モードの選択

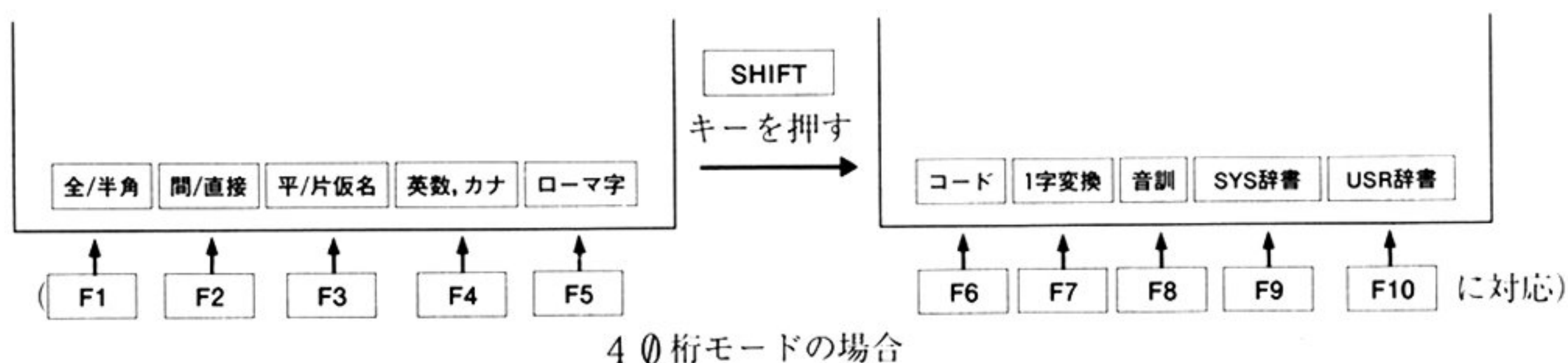
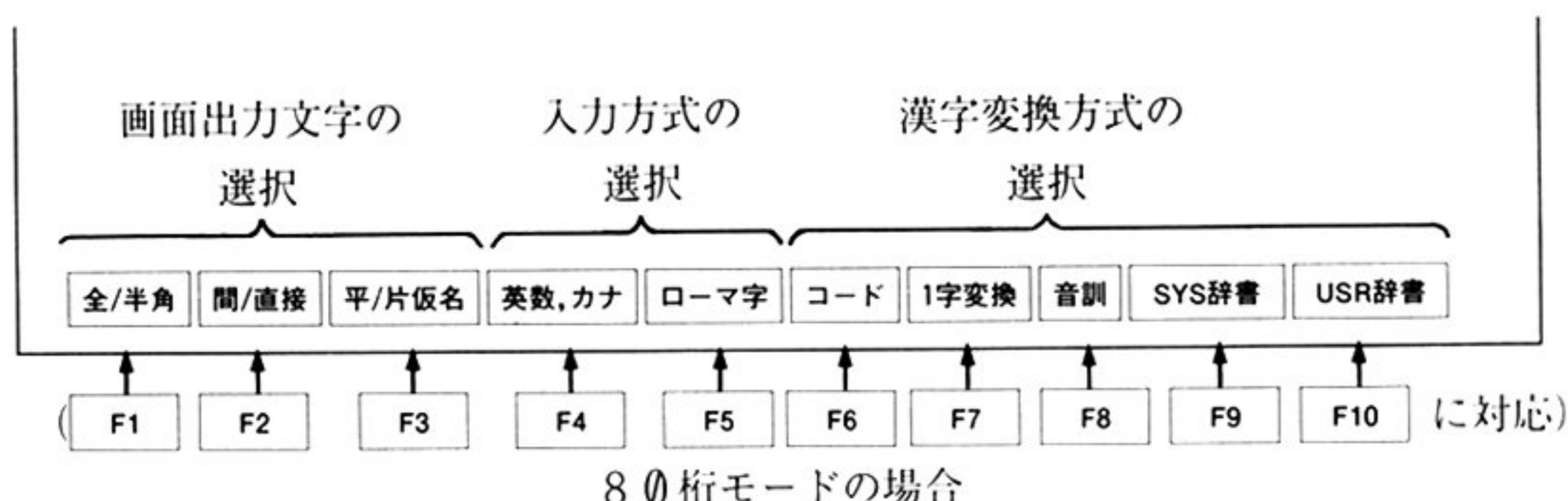
日本語入力モードでは、画面に出力される文字、入力方式、漢字変換方式をファンクションキー

**F1** ～ **F10** によって選択することができます。



### 3.1 HELP キーについて

SHIFT + XFER または CTRL + XFER で日本語入力モードにし、次に HELP キーを押してください。すると最下行の変換フィールドは次のように変わります。



このように、入力モードの確認は HELP キーによって行ないます。上記の画面で、**現在のモードは赤字**で表示されます。(各モードについては次の3.2で説明します) これらのリストを表示したままの状態、F1 ~ F10 キーを押せば、入力モードの変更ができます。日本語入力モードに戻るためには、再び HELP キーを押します。

### 3.2 各入力モードの選択

(1)画面出力文字の選択 (F1 , F2 , F3 について各々選択)

- F1 : 全角／半角……………全角文字か半角文字のどちらを表示させるかを選択します。
- F2 : 直接／間接……………直接、テキストエリアのカーソル位置に出力するか、またはいったん変換フィールド内に出力するかを選択します。
- F3 : 平仮名／片仮名……ローマ字→カナ変換方式またはカナ入力方式で入力した時に、ひらがなを出力するか、カタカナを出力するかを選択します。

(2)入力方式の選択 (F4 , F5 , カナ の中から1つ選択)

- F4 : 英数字直接入力方式……キーボード上のアルファベット、数字、記号の通りに入力します。この入力方式を選択すると、自動的に画面出力文字も英数字、記号となります。



**F5** :ローマ字—カナ変換方式……アルファベットを入力すると、自動的にローマ字とみなされ、ひらがなまたはカタカナに変換されます。

※ **カナ** キーロックによるカナ入力方式

入力方式が **F4** の英数字または **F5** のローマ字—カナ変換のモードにある時、**カナ** キーをロック（押し下げた状態）すれば、直接カナによる入力ができます。

(3)漢字変換方式の選択 (**F6**, **F7**, **F8**, **F9**, **F10** の中から1つ選択)

**F6** :コード入力方式……全角文字のコードを直接入力します。コードにはJIS漢字コード (**SHIFT** + **F1**) ド ("16進"と表示)と区点コード ("区点"と表示)とがあり、このキーを押す毎に切替わります。

**F7** :一字変換方式……漢字の読みを1文字だけ識別 (他の文字は無視)して漢字に変換 (**SHIFT** + **F2**) します。

**F8** :音訓変換方式……漢字の音読み、訓読みを入力して、漢字に変換します。  
(**SHIFT** + **F3**)

**F9** :システム辞書変換方式…日本語百科ワードパワー (WORD POWER)ディスクや (**SHIFT** + **F4**) システム辞書ディスク (オプション: CZ-111SF)を使用するときの方式です。

**F10** :ユーザー辞書変換方式……ターボ博士レキシコン (LEXICON)ディスクや (**SHIFT** + **F5**) ユーザー辞書ディスク (オプション: CZ-111SF)を使用するときの方式です。

(4)一字変換、音訓変換、システム辞書・ユーザー辞書変換方式について

①一字変換機能 ……コンピュータの内部プログラムとして常駐しています。

②音訓変換方式 ……これを利用するためには、音訓辞書が必要です。音訓辞書はシステムディスクの中にあります。

③システム辞書、ユーザー辞書変換方式…それぞれWORD POWERディスク、LEXICONディスクまたはシステム・ユーザー辞書 (オプション: CZ-111SF)が必要です。

なお②、③の変換方式には、変換エリアをメモリ内部に確保するために、LIMITコマンド (またはCLEARコマンド)でエリアの確保を行なう必要があります。各方式におけるエリア領域は次のとおりです。

LIMIT & HF 000 (または CLEAR & HF 000)

※入力モードの選択はファンクションキー **F1** ~ **F10** のかわりに **GRAPH** キーを押しながら数字キー **1<sub>α</sub>** ~ **0<sub>7</sub>** を入力することによって行なうことができます。

(ただし、数字キーとしてテンキーを用いることはできません)

[例] ○全角／半角の切換え方法… **F1** または **GRAPH** + **1<sub>α</sub>** を入力する。

○音訓変換方式への切換え方法… **F8** (**SHIFT** + **F3**) または **GRAPH** + **3<sub>ア</sub>** を入力する。

漢字変換方式には前項でも説明したように、次の5種類があります。

1. コード入力方式
2. 一字変換方式
3. 音訓変換方式
4. システム辞書変換方式
5. ユーザー辞書変換方式

1,2,3について説明します。4,5については『ターボ博士レキシコン、日本語百科ワードパワーの説明書』またはオプションのシステム・ユーザー辞書(CZ-111SF)の説明書をお読みください。

なお、上記1～5の漢字変換方式を選択するには次のように行ってください。

①ディスクBASIC (CZ-8FB02) を起動します。

②次に、日本語入力モードに入ります。

(**SHIFT** + **XFER** または **CTRL** + **XFER** をキー入力します)

③ **HELP** キーを押すと、

漢字変換方式の 選択									
全/半角	間/直接	平/片仮名	英数、カナ	ローマ字	コード	1字変換	音訓	SYS辞書	USR辞書

**BASIC** 起動時は音訓変換方式になっています。

- ・コード入力方式を選択するときは、
- ・一字変換方式を選択するときは、
- ・音訓変換方式を選択するときは、
- ・システム辞書変換方式を選択するときは、  
ドライブ0にWORD POWERまたはシステム辞書  
ディスクを入れて
- ・ユーザー辞書変換方式を選択するときは、  
ドライブ0にLEXICONまたはユーザー辞書  
ディスクを入れて

**F6** (**SHIFT** + **F1**)

**F7** (**SHIFT** + **F2**)

**F8** (**SHIFT** + **F3**)

**F9** (**SHIFT** + **F4**)

**F10** (**SHIFT** + **F5**)

のキーを押してください。

④次に **HELP** キーを押すと、日本語入力モードに戻っています。

[平仮名] ■	全/半角	間/直接	ローマ字	1字変換
---------	------	------	------	------

一字変換入力方式にした例

## 4.1 コード入力方式

○コード入力方式を選択するには

**F6** (**SHIFT** + **F1** キー) または **GRAPH** + **6** キーを入力します。



○コードには **J I S 漢字コード**と**区点コード**の2種類がありどちらで入力するかは同じく

**F6** (**SHIFT** + **F1** キー) または **GRAPH** + **オ** キー

を入力することによって切換えることができます。

**J I S 漢字コード**が選択されると **〔16進〕**

**区点コード**が選択されると **〔区点〕**

の文字が表示されます。

○J I S 漢字コードと区点コードの間には次式で示すような関係があります。ただしJ I S 漢字コードは16進数、区点コードは10進数で表わします。

$$\begin{cases} \text{区点コード (上位バイト)} = (\text{J I S 漢字コード (上位バイト)} - 20H)_{10} \\ \text{区点コード (下位バイト)} = (\text{J I S 漢字コード (下位バイト)} - 20H)_{10} \\ \text{J I S 漢字コード (上位バイト)} = (\text{区点コード (上位バイト)} + 32)_{16} \\ \text{J I S 漢字コード (下位バイト)} = (\text{区点コード (下位バイト)} + 32)_{16} \end{cases}$$

○コード入力方式が選択されると、変換フィールド内には**数字の0～9およびアルファベット大文字のA～Fの入力以外受けつけられません。**

○コードの入力は**4桁**で行ない (上位2桁：上位バイト、下位2桁：下位バイト)

**J I S 漢字コード**選択時には **16進数**

**区点コード**選択時には **10進数**

とみなされます。

○また、コードと漢字等の全角文字とは1対1に対応しており、その対応関係の詳細は別表に示される通りです。以下にその概略を示します。

JIS 漢字コード (16進数)	区点コード (10進数)	対応文字 (全角文字)
0021～207E	0001～0094	入力無効 (無表示)
2121～2F7E	0101～1594	第1水準 非漢字
3021～4F7E	1601～4794	第1水準 漢字
5021～757E	4801～8594	第2水準 漢字 (オプション)
7621～7660	8601～8664	外字 (PCG 登録文字)

(注) 以下のコードを入力しても無効となります。(※……任意の数)

J I S 漢字コード：※※00～※※20、※※7F～※※FF

区点コード：※※00、※※95～※※99

(注) コード入力方式では、**XFER** キーを押す必要はありません。4桁のコードを入力した時点で、全角文字に変換されてしまいます。

〔例1〕変換フィールド内でJ I S 漢字コード3021を入力すると、テキストエリア内に漢字"亜"が表示されます。

亜■ 〔16進〕■	全/半角 間/直接 ローマ字 コードIN
--------------	----------------------

〔例2〕変換フィールド内で区点コード1601を入力すると、テキストエリア内に漢字"亜"が表示されます。



■					
[区点]		全/半角	間/直接	ローマ字	コードIN

○全角文字には、漢字、非漢字文字以外に、**ユーザー登録文字（外字）**があります。

外字はPCGを利用しており、次の2種類があります。

外字全角文字…PCG 4 キャラクタ分に定義

外字半角文字…PCG 2 キャラクタ分に定義

（外字の定義方法については『アプリケーションソフトの説明書』の「デフチャーツール」の項を参照してください。）

ここでは**外字全角文字の表示方法**についてのべます。

コード入力方式では、外字全角文字を表示させるために、外字の各文字にコードを割り当てています。外字用のコードは、第1水準および第2水準の全角文字と衝突しないように、次のように割り当てられています。

JIS漢字コード：7621～7660（64文字分）

区点コード：8601～8664（64文字分）

したがって、外字全角文字を表示させるためには上記コードのいずれかをキー入力します。

上記コードのどれがPCGのどのキャラクタに対応しているかについては、『アプリケーションソフトの説明書』の「デフチャーツール」を参照してください。

〈注意〉

外字半角文字は、コード入力方式では表示できません。外字半角文字を表示するためには次のようにします。

CGEN2

PRINT#0, CHR\$(X)

ただし、Xは0から&HFEまでの偶数のPCGキャラクタコードです。

## 4.2 一字変換方式


カナ→漢字変換方式の一つで、変換フィールド内に入力したカナの最初の1文字を判断して、先頭にその読みを持つ漢字を、JIS漢字コードの順番通りに羅列します。羅列の仕方は、9文字を一つの漢字グループとみなして1度に最大9文字ずつ、該当する漢字がなくなるまで交代に表示します。

(1)変換例 一字変換方式によって、漢字"日本"を表示させます。

①入力モードを全角文字出力（F1）、間接出力（F2）、平仮名出力（F3）ローマ字入力（F5）、一字変換方式（F7）に設定します。

②変換フィールドにN I C H I HTAB H O X  
の各キーを入力します。


■					
[平仮名]	に	ち	い	ほん	■
		←	全/半角	間/直接	ローマ字
					1字変換

③ **XFER** キーを押します。すると、" にち " の先頭文字 " に " の読みを頭を持つ漢字が羅列されます。さらに、目的の漢字が見つかるまで **XFER** キーかカーソルコントロールキー  かスペースキーを押し続けます。

■  
〔平仮名〕 にち  尼式 邇匂賑 肉虹甘

■  
〔平仮名〕 にち  乳入 如尿蕐 任妊忍

④ 目的の漢字 " 日 " を取り出します。(  キーを押すか、数字キー **1** を押します)

日 ■  
〔平仮名〕 ほん ■  **全／半角** **間／直接** **ローマ字** **1 字変換**

⑤ 同様に、 **XFER** キーを押してカナ " ほん " の先頭文字 " ほ " の読みを頭にもつ漢字を羅列します。

〔平仮名〕 ほん

保舗舗	圃捕歩	甫補輔		キー入力
穂募墓	慕戊暮	母簿菩		
倣俸包	呆報奉	宝峰峯		
崩庖抱	捧放方	朋法泡		キー入力
烹咆縫	胞芳萌	蓬蜂褒		
訪豊邦	鋒飽鳳	鵬乏亡		
傍剖坊	妨帽忘	忙房暴		キー入力
望某棒	冒紡肪	膨謀貌		
質銓防	吠頰北	僕卜墨		
撲朴牧	睦穆鉤	勃沒殆		キー入力
掘幌奔	本翻凡	盆		

日 ■  
〔平仮名〕 ほん



⑥目的の漢字"本"を取り出します。(数字キー $\boxed{4}$ を押します)

日本■			
[平仮名] ■	全/半角	間/直接	ローマ字 1字変換

以上で 漢字"日本"が入力されます。

(2)ところで、一字変換方式にはもう1つ別の変換の仕方があります。それは、変換フィールドに入力したキャラクタの先頭の1文字を判断して、そのキャラクタコードをJIS漢字コードの上位バイトに対応させた場合の該当する全角文字を羅列する方式です。たとえば、キャラクタ"!"(キャラクタコード:&H21)を入力した場合、変換フィールドエリアに羅列させる全角文字は、JIS漢字コード2121~217Eまでの94個の文字群になります。この方式ですと、入力文字のキャラクタコードがそのままJIS漢字コードの上位バイトに対応しますので、変換フィールド内に入力するキャラクタは"!"(キャラクタコード:&H21)から"u"(キャラクタコード:&H75)までの85文字ということになります。以下に、入力キャラクタと、対応するJIS漢字コードとの関係を簡単にまとめます。

入力キャラクタ	キャラクタコード	JIS漢字コード	補 足
! ~ /	&H21 ~ &H2F	2121 ~ 2F7E	第1水準 非漢字
0 ~ O	&H30 ~ &H4F	3021 ~ 4F7E	第1水準 漢字
P ~ u	&H50 ~ &H75	5021 ~ 757E	第2水準 漢字(オプション)

羅列された漢字グループから目的の漢字を取り出す方法は、通常的方式とまったく同じです。詳しくは、『入門編』の「5章 漢字を表示する」の項を参照してください。

(注)入力方式としてローマ字—カナ変換方式またはカナ入力方式を選択している場合は、アルファベットや記号の一部を入力することができません。その場合は、英数字直接入力方式に切替えてください。

### 4.3 音訓変換方式

カナ—漢字変換方式の1つで、変換フィールド内に漢字の音読みまたは訓読みを入力して、それを音訓辞書を利用することによって全角文字に変換します。ディスクBASICを起動させたとき、日本語入力モードにすると音訓変換方式が初期状態として設定されます。

■			
[平仮名] ■	←全/半角	間/直接	ローマ字 音訓

この音訓変換方式の利用については入門編「5章 漢字を表示する」で説明していますので、参照してください。



ところで、音訓変換方式には、漢字の音読み、訓読みだけでなく、各種記号、ギリシア文字、ロシア文字、外字全角文字を表示させるために、次の4種類のカナ文字を入力することができます。

- (1) "キゴウ" .....各種記号を羅列します。
- (2) "ギリシア" .....各種ギリシア文字を羅列します。
- (3) "ロシア" .....各種ロシア文字を羅列します。
- (4) "ガイジ" .....外字全角文字を羅列します。

変換方法は簡単です。すなわち、変換フィールド内に上記カナ文字列を入力した後 **XFER** キーを押します。すると、変換フィールドエリアに各種全角文字が羅列されますので、**入門編「5章 漢字を表示する」**の項で述べたように、カーソルコントロールキーや数字キーによって目的の文字列を取り出すことができます。

#### ※ 音訓辞書の学習機能について

カナ→漢字変換の音訓変換方式では、使用した漢字を同一グループ内の先頭に配置変換する機能をもっています。したがって、直前に使用した漢字を先頭に羅列します。

この機能は、数字キーによる漢字の選択を行なった場合にのみ行なわれ、カーソルコントロールキーによる選択時には行なわれません。また、同梱の音訓辞書ディスク（マスターディスク）が書き込み禁止状態にある時には働きません。

#### ※ 辞書機能を利用する前に...

音訓辞書機能やシステム辞書機能、ユーザー辞書機能を利用して、漢字変換を行なうためには、それぞれ音訓辞書、システム辞書ディスク（オプション：C Z-1 1 1 S F）、ユーザー辞書ディスク（オプション：C Z-1 1 1 S F）が必要です。これらの辞書機能を利用する場合は、辞書ディスクを挿入したデバイス名をコンピュータ側に教えなければなりません。そのためには、次の操作を行ないます。

- ① D E V I C E コマンドを使って、辞書ディスクを挿入したデバイス名をデフォルトデバイス<sup>(1)</sup>として指定します。

- ・フロッピーディスク... ① : ~ 3 :
- ・グラフィックメモリ... M E M ① : , M E M 1 :

- ② 日本語入力モードにして、漢字変換方式を選択します。選択の仕方は次の通りです。


- ・音訓辞書機能を利用する場合 : **SHIFT** + **F3** キー入力
- ・システム辞書機能を利用する場合 : **SHIFT** + **F4** キー入力
- ・ユーザー辞書機能を利用する場合 : **SHIFT** + **F5** キー入力




このキーを入力することによって、コンピュータ側はそれぞれ辞書機能を利用する際のアクセスを知ります。



(注) デフォルトデバイス... デバイス名の指定を省略したときに用いられるデバイス

以上の操作によって、コンピュータはそれ以後の辞書ディスクをアクセスする際には、指定したデバイスに対してアクセスを行ないます。一度この操作を行なうと、再度同様の操作によって指定を変更しない限り、アクセス先は変わりません。したがって、上記②のキー入力を行なうときは、デフォルトデバイスに何が指定されているのか注意する必要があります。また、各辞書機能を並行して使用する場合は、利用する辞書を変更するたびに上記①、②の操作をくり返してください。

〔例〕 音訓辞書ディスクをドライブ1に挿入して利用する場合の操作方法

① DEVICE "1:"  ..... デフォルト・デバイスの指定

②  +  ..... 日本語入力モードに切換え。詳しくは、本章「 全角文字の入力方式」を参照してください。

③  +  ..... 音訓変換方式の選択

※デバイスを変更しても、音訓辞書の読み出されるデバイスは、変化しません。

以上の操作を応用すれば、音訓辞書的高速利用が可能になります。これは、音訓辞書をそっくりそのままグラフィック用V-RAMに転送することによって行ないます。より高速な音訓変換を行ないたい場合などに用いられます。アクセス速度は、フロッピーディスクよりもグラフィック用V-RAMの方が高速ですので、グラフィック用V-RAM上に音訓辞書を移せば、音訓変換を一瞬のうちに行なうことができます。そのためには、次の操作を行ないます。


① グラフィック用V-RAMの使用目的を、外部記憶用に設定します。

OPTION SCREEN 2 

"OPTION SCREEN 3"または"OPTION SCREEN 4"でもかまいません。(詳しくは、『BASICリファレンス・マニュアル』の「OPTION SCREEN」の項参照。)

② グラフィック用V-RAMを初期化します。

INIT "MEM0:" 

とすると、次のメッセージが表示されますのでキーを押します。

Are you sure ? (y or n)

("OPTION SCREEN 3"に設定した時は、"MEM0:"のかわりに"MEM1:"を実行してください(以後、同じようにします。))



③ 音訓辞書(ドライブ0)をグラフィック用V-RAMに転送します。

COPY "0:音訓変換.DIC" AS "MEM0:" 



④ デフォルト・デバイスをグラフィック用V-RAMに指定します。

DEVICE "MEM0:" 

⑤ 日本語入力モードに切換えます。

 + 

⑥ 音訓変換方式を選択します。

 + 

⑦ デバイスの指定を④以前にもどします。



以上で、以後の音訓変換はグラフィック用V-RAMに対してアクセスされますので、非常に高速な漢字変換が行なえます。

注) システム辞書及びユーザー辞書については、グラフィック用V-RAMを利用して高速変換を行なうことはできません。




## 操作キー一覧






### ●日本語入力モードにおける特殊キーの動き

キー入力	動作
<div>SHIFT + XFER</div> <div>または CTRL + XFER</div>	日本語入力モードに切り換えたり、日本語入力モードから抜け出したりする。
<div>XFER</div>	変換を行なう。
<div>HELP</div>	入力モードの確認。
<div>F1 ~ F10</div> <div>または GRAPH +  ~ </div>	入力モードの選択。

### ●変換フィールドエリア内に入力された状態での働き

キー入力	動作
	変換フィールドエリア内の文字をテキストエリアに移す。
<div>INS DEL</div>	変換フィールドエリア内の文字を抹消する。
<div>CLR HOME</div>	変換フィールドエリア内の文字をクリアする。
カーソルコントロールキー	無効。
<div>HTAB</div>	連続して漢字を変換する。

### ●漢字グループを表示している状態

キー入力	動作
<div></div> <div></div>	漢字グループ内のカーソルを移動する。
<div></div> <div></div>	漢字グループの次候補、前候補を選択する。
<div>ESC</div>	漢字グループの表示を消し、変換フィールド内にカーソルを戻す。
	目的の漢字をテキストエリアに表示する。
テンキー <div>1</div> ~ <div>9</div>	目的の漢字に対応したテンキーを押せばその漢字をテキストエリアに表示する。



# 6章

## フリーエリアについて

本機では、CPUにZ80Aを採用しています。したがって、メインメモリは64Kバイトのメモリ空間を持っています。このうち、BASICインタープリタに使用している領域を除いた残りの部分がフリーエリアになります。

このフリーエリアの領域にプログラムやデータが格納されるわけですが、それらがフリーエリアより大きくなるとメモリが足りなくなり、"Out of memory"エラーになります。

したがって、大きなプログラムを走らせたり、多量のデータを扱うためには、大きなフリーエリアを確保することが必要ですがメインメモリは64Kバイトと制限されていますので、この中でフリーエリアを増やそうとすると、BASICを小さくするしかありません。

しかし、BASICを小さくすると、その分機能が減り、かえって使いにくいものになってしまいます。

そこで、本機ではグラフィックVRAMを変数領域として使用できるようにし、その分メインメモリのプログラム領域を増やし、フリーエリアの増大を図っています。

### 1

### フリーエリア

本機のメモリはBIOS ROM、メインメモリ、グラフィックVRAMなどから構成されています。

BASICは、このうちのメインメモリにロードされ、残りの領域にプログラムテキストやプログラムに使用される変数が置かれ通常この部分をフリーエリアといいます。

しかし、フリーエリアは、プログラムの実行前と実行後やグラフィックVRAMを変数に使うか否かなどによって異なります。

そこで、もう少しフリーエリアについて詳しく見ていくことにします。

#### 1.1 BASICの起動直後

BASICを起動した直後はメインメモリにはBASICシステムとそれに必要なワークエリアなどがとられるだけです。また、グラフィックVRAMは変数として使用されるモードとなっていますので、図—1のようになります。

#### 1.2 プログラムロード後（実行前）

プログラムをロードするとBASICシステムの後ろにプログラムテキストが置かれます。したがって、フリーエリアはメインメモリからBASICシステムワークエリア、プログラムテキストを除いた部分と、グラフィックVRAMの部分で図—2のようになります。

### 1.3 プログラム実行後

プログラムを実行するとプログラムテキストの後ろにプログラムで使われる変数がとられます。またVDIM命令を使っていれば、グラフィックVRAMにも変数がとられます。

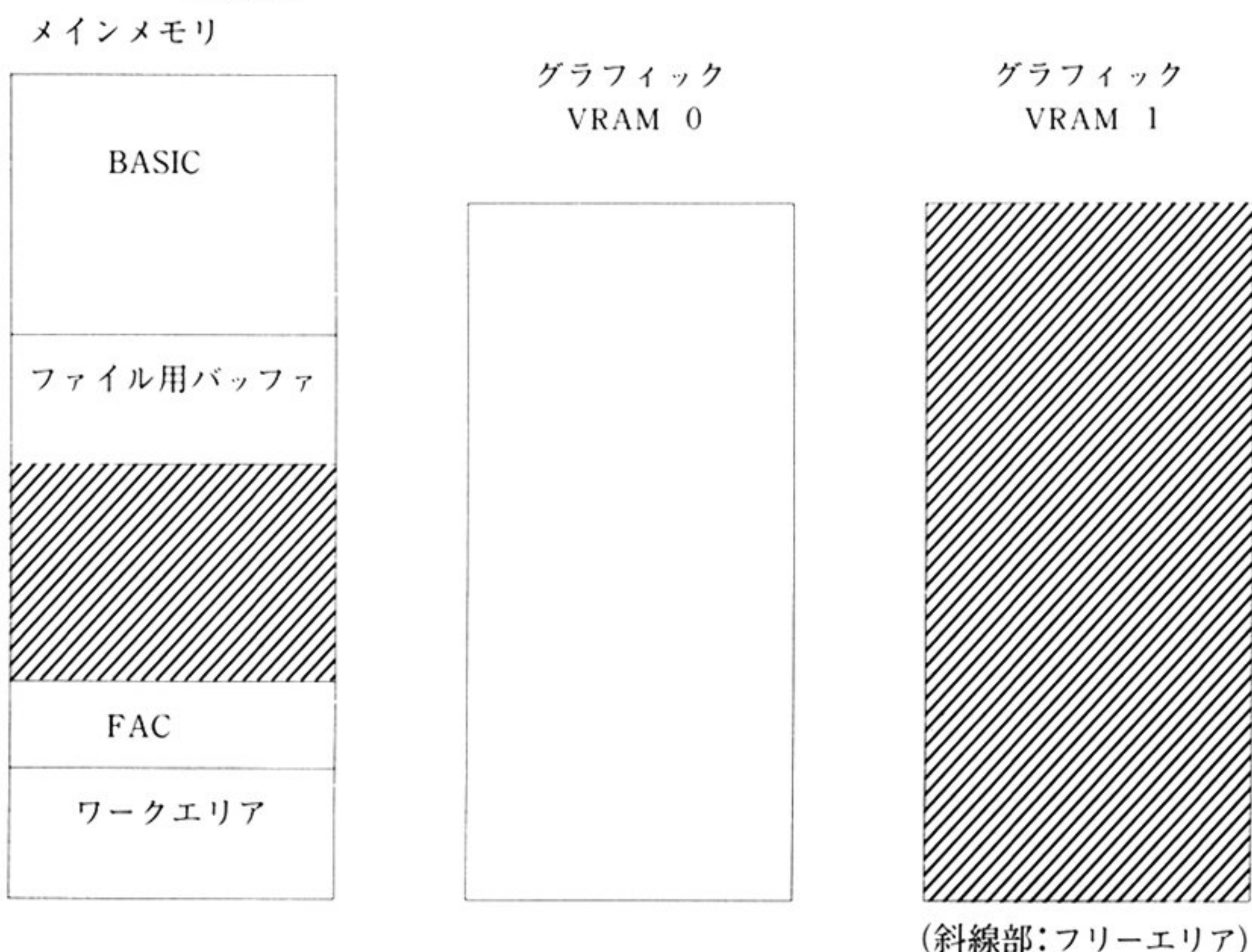
この場合のフリーエリアは図-3で示すようにプログラムロードの後のフリーエリアからさらに変数エリアを除いた部分になります。

### 1.4 グラフィックVRAMを変数として使用しない場合

高解像度グラフィックを使用する場合、グラフィックVRAMのバンク1は、グラフィック描画用に使用され、変数を格納することはできません。また、外部記憶として使用する場合も同様に変数を格納することはできません。

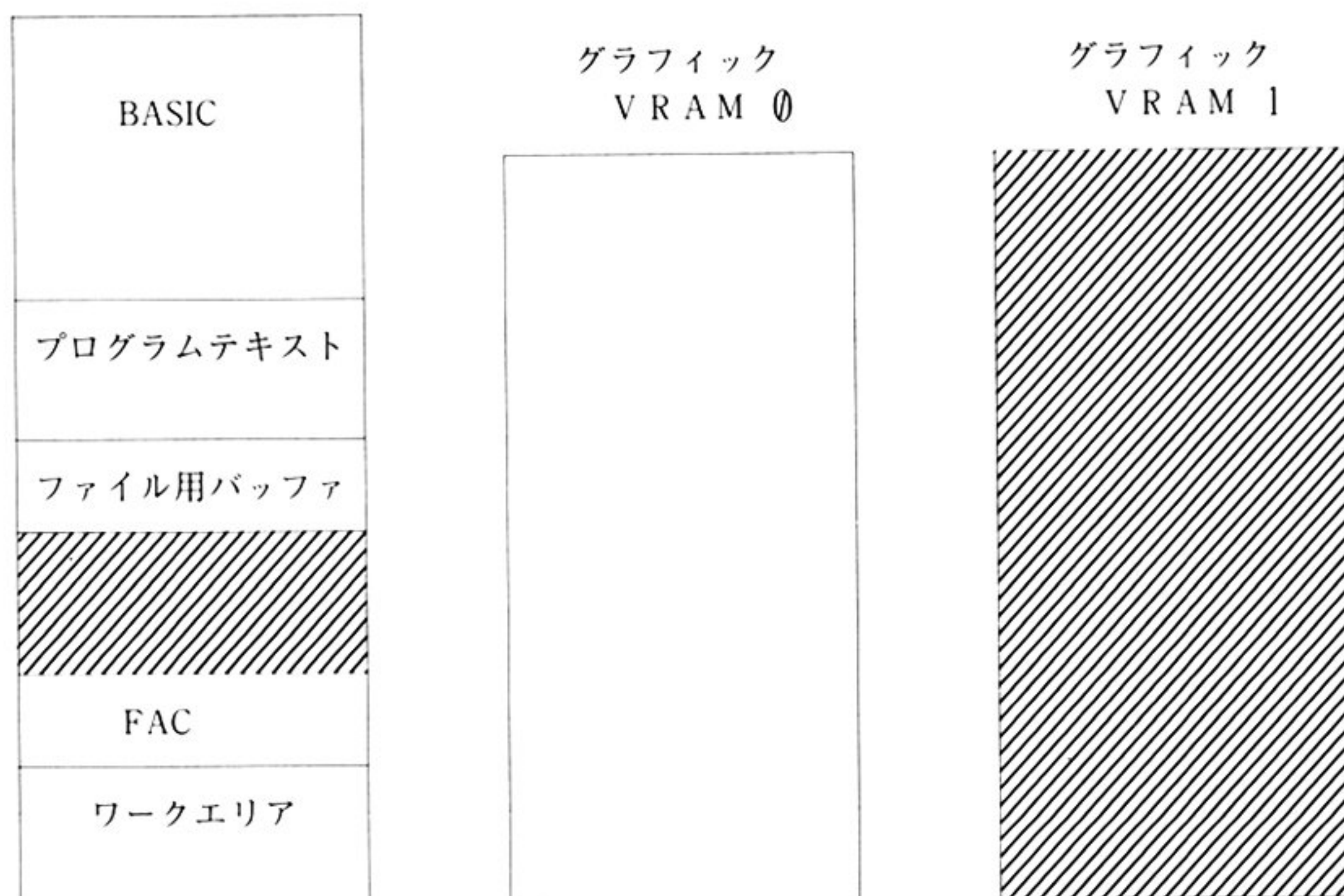
これらの場合、メインメモリの部分のみがフリーエリアとなります。

図-1 BASIC起動直後

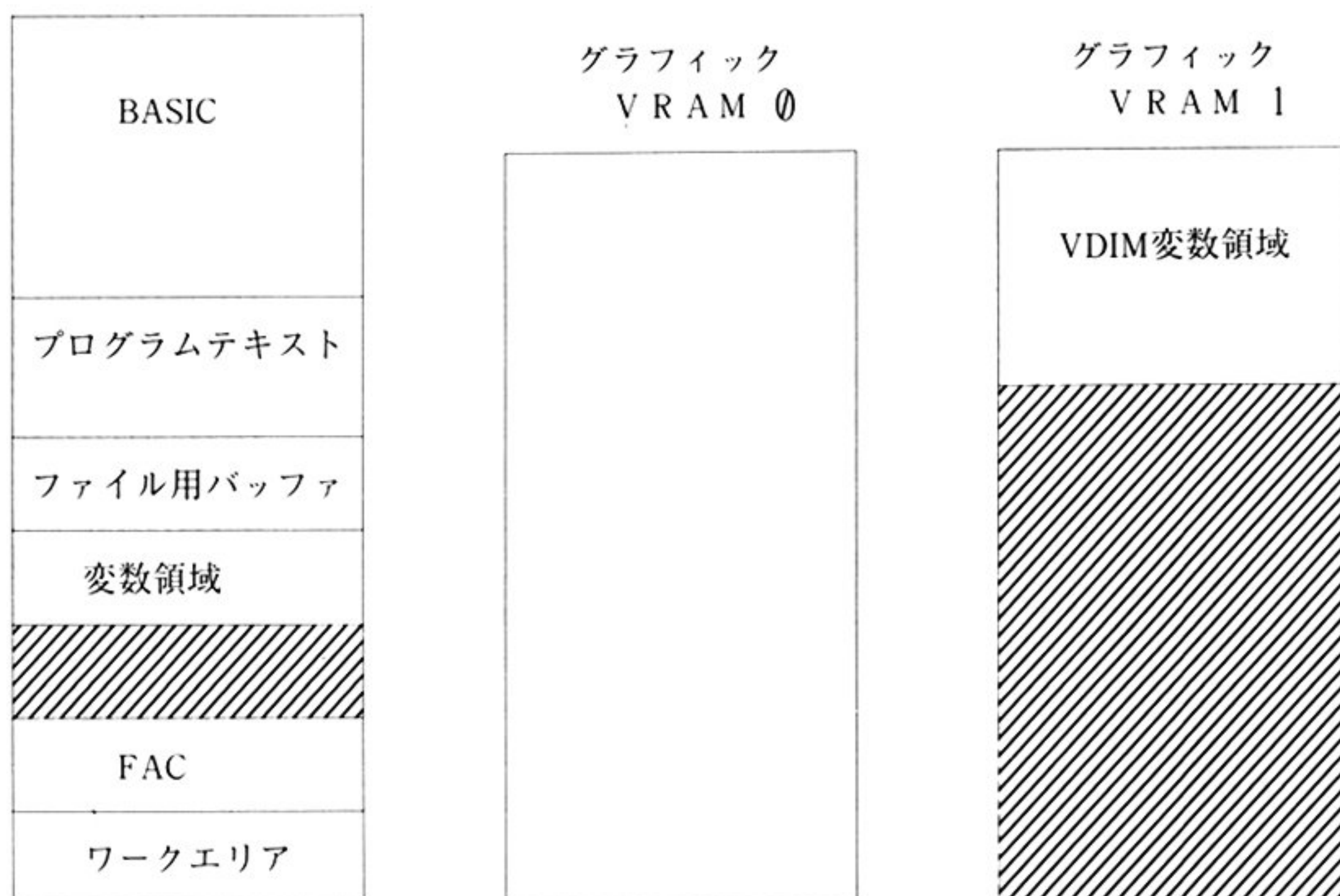




図ー2 プログラムロード後（実行前）  
メインメモリ



図ー3 プログラム実行後  
メインメモリ



- F A Cとは演算処理時に使用されるワークエリアです。
- ファイル用バッファはファイルをアクセスする際に使用されるバッファでMAXFILE S命令により確保される大きさが異なります。

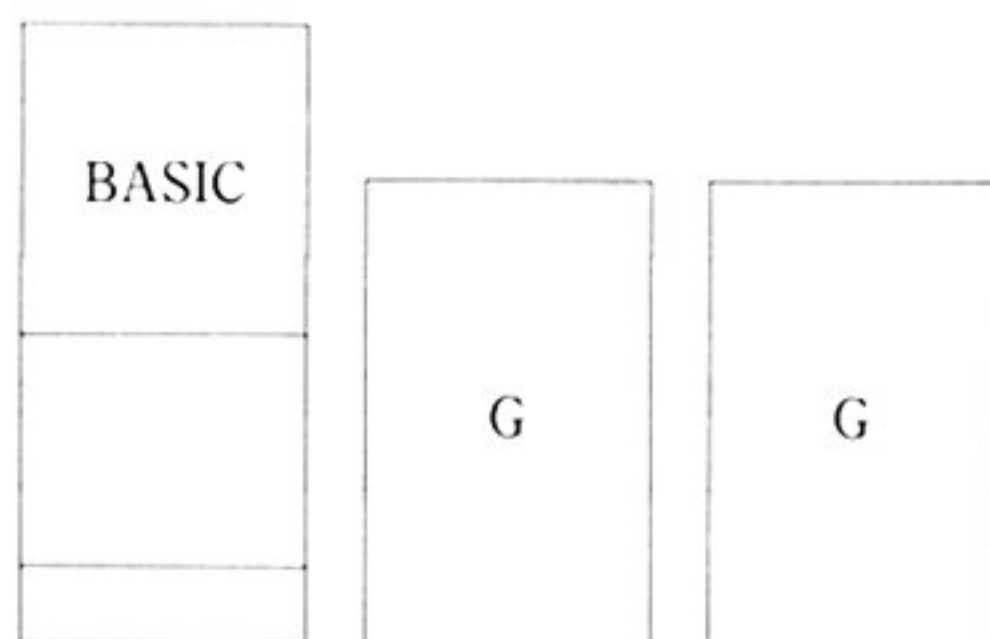


グラフィックVRAMをどのような用途で使用するかを決めるにはOPTIQN SCREENという命令を使います。

OPTION SCREENで定義されるスクリーンモードは全部で5つあり、OPTION SCREEN 0～OPTION SCREEN 4で設定します。

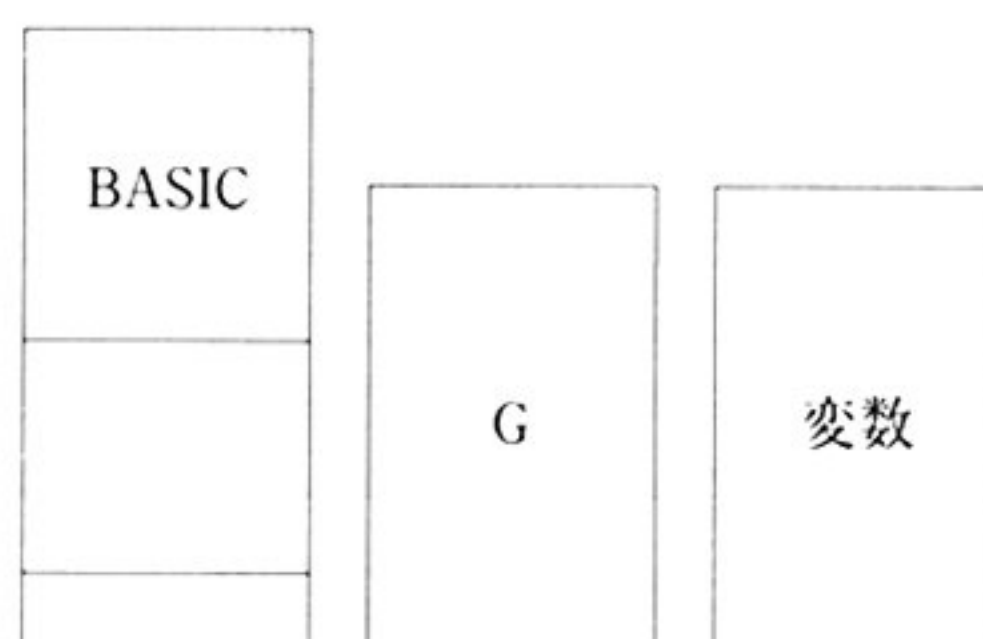
このうち、グラフィックVRAMを変数領域として使用できるのはOPTION SCREEN 1とOPTION SCREEN 2で、それ以外はすべてグラフィックまたは外部記憶とします。

(1)OPTION SCREEN 0

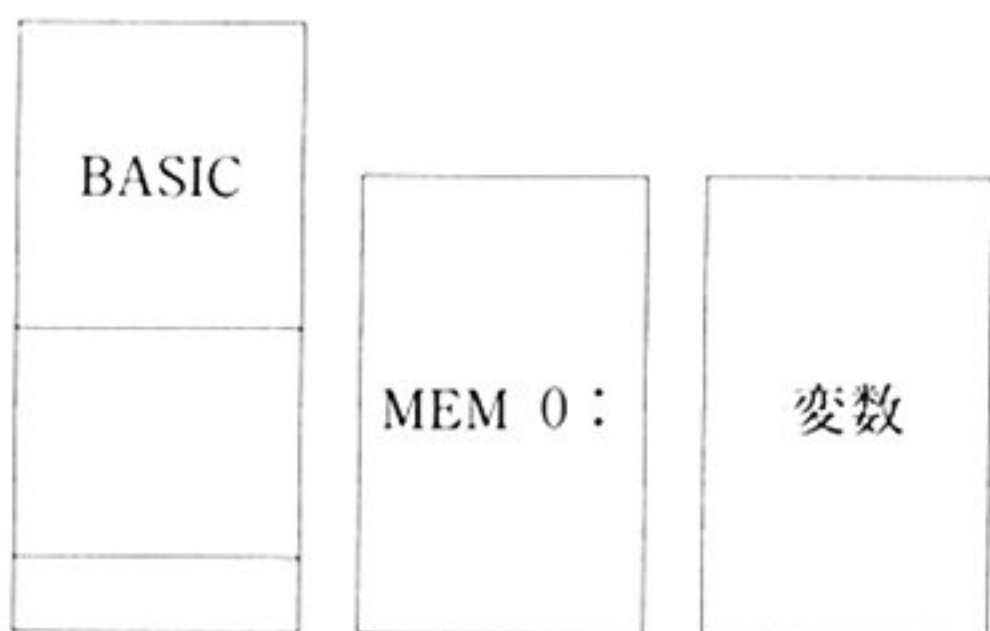


(2)OPTION SCREEN 1

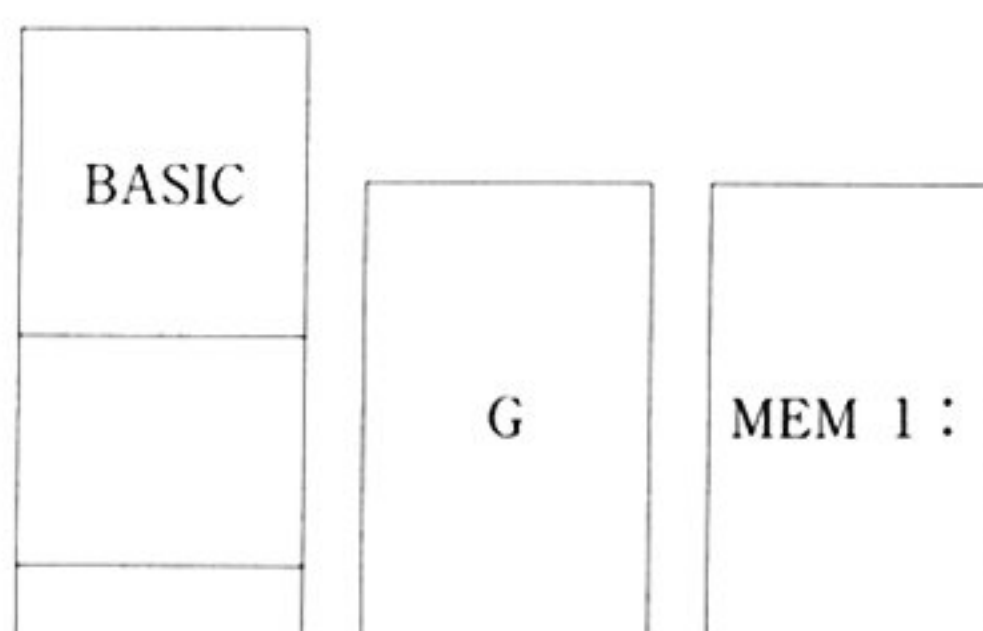
(BASIC起動時の初期状態)



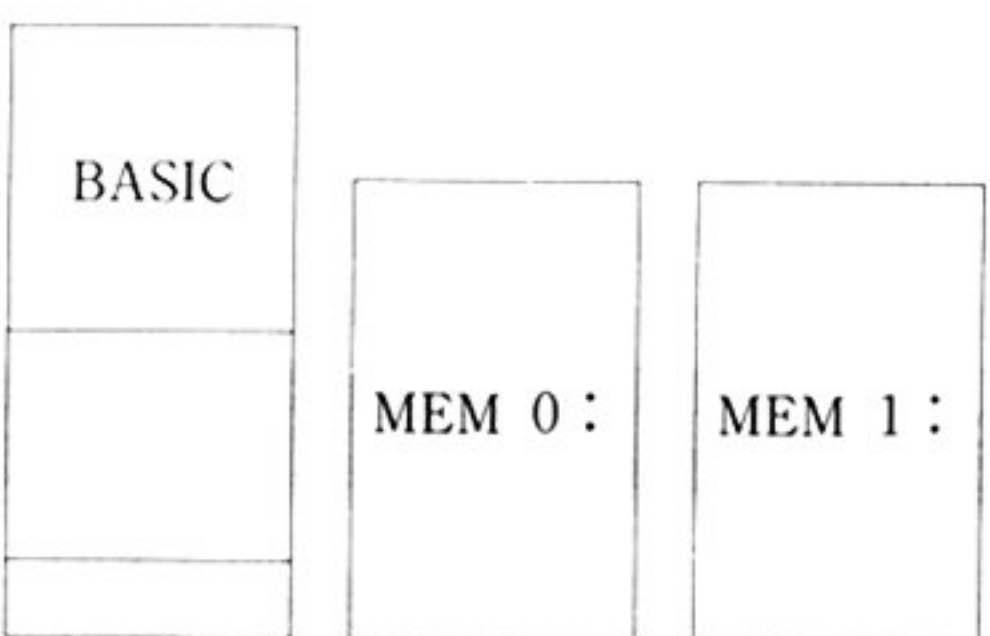
(3)OPTION SCREEN 2



(4)OPTION SCREEN 3



(5)OPTION SCREEN 4



Gはグラフィック描画用

MEM 0:, MEM 1:はデバイス名を示します。

(1)OPTION SCREEN 0

グラフィックVRAMのバンク0、バンク1ともグラフィックの描画に使用されます。

その結果、グラフィックとしては次の画面モードで使用できます。

640×400	1画面	320×400	2画面
640×384	1画面	320×384	2画面
640×200	2画面	320×200	4画面
640×192	2画面	320×192	4画面

このモードではVDIM、VDIM CLEAR、INIT"MEM0："、INIT"MEM1："の命令を実行しようとするとき"Bad screen mode"エラーとなります。さらに、640×400、320×400、640×384、320×384の各画面モードにおいては次のステートメントを実行することができません。

```
OPTION SCREEN 1
OPTION SCREEN 2
```

また、変数、プログラム領域として使用できるのはメインメモリ上のみです。

#### (2)OPTION SCREEN 1

グラフィックVRAMのバンク0は、グラフィック描画用として使用し、バンク1を変数領域として使用します。

画面モードとしては、

640×200	1画面	320×200	2画面
640×192	1画面	320×192	2画面

となり、スクリーンの解像度は制限されますが、VDIM命令によりグラフィックエリアに変数を確保することができます。

このモードでのフリーエリアはメインメモリ上の領域とグラフィックVRAMのバンク1を合わせた部分となります。

#### (3)OPTION SCREEN 2

グラフィックVRAMのバンク0は外部記憶として使用し、バンク1を変数領域として使用します。

したがって、フリーエリアはメインメモリ上の領域とグラフィックVRAMのバンク1を合わせた部分となります。

このモードではVDIM、VDIM CLEAR、INIT"MEM0："は使用できますが、INIT"MEM1："とすると"Bad screen mode"のエラーとなります。

#### (4)OPTION SCREEN 3

グラフィックVRAMのバンク0はグラフィック描画用として使用し、バンク1を外部記憶として使用するモードです。

プログラムテキスト、変数は共にメインメモリ上にのみ確保されます。

#### (5)OPTION SCREEN 4

グラフィックVRAMのバンク0、バンク1共に外部記憶用として用います。

プログラムテキスト、変数は共にメインメモリ上にのみ確保されます。

## 3

## グラフィック描画と外部記憶

OPTION SCREEN 2、3、4のモードではグラフィックVRAMを外部記憶として使用できますが、このメモリ上にグラフィックデータを描画することも可能です。

すなわち、次の点に注意すれば、グラフィックVRAMを外部記憶として設定した場合でも、グラフィックを描画することもできます。

また、48Kバイトのうち一部を外部記憶として、ファイルを格納し、残りをグラフィック描画にすることも可能となります。

(1)グラフィックVRAMを外部記憶として、設定した場合、WIDTH命令でメモリの内容がクリアされることはありません。



(2)グラフィックVRAMに対し、INIT命令を実行すると、グラフィックVRAMにはメモリ管理のためのデータが書き込まれます。

(3)外部記憶として、使用した場合でも、グラフィック命令によって、グラフィックを描画できますので、ファイルと共存して使用する場合は注意が必要です。

たとえばCLS命令などによってファイルをこわさないようにしてください。

ファイルはグラフィックVRAMの上位アドレス（青のメモリ）から順に書き込まれていきますので、上位16Kバイトにファイルを格納し、残り32Kバイト（赤と緑）をグラフィックに使うということも可能です。

## 4

## 日本語入力とフリーエリア

ディスクBASICを起動した時点では、日本語入力は音訓入力モードになっています。音訓入力モードでは音訓辞書ディスクから読み込んで変換を行ないますので、音訓辞書をアクセスするためのルーチンが必要です。

このルーチンはメインメモリ上のF000H番地からF3FFH番地を使用するため、

CLEAR & HF000

となっています。

したがって、音訓変換を使用しない場合には、

CLEAR & HF400

とすることができますので、フリーエリアが1Kバイト増加します。

## 5

## NEWONとフリーエリア

本機のBASICはゲーム、ビジネス、教育用などあらゆる用途に対応できるため、ひじょうに多くの命令を含んでいます。

その結果、短いステップで豊富な処理を扱うことができます。

しかし、実際のプログラムではすべての命令を使用しているわけではありません。

たとえば、外部デバイスにディスクしか使わないプログラムではカセットの命令やRS-232Cの命令は不要です。それらの命令に使用しているメモリ領域をユーザー用のフリーエリアとすればより大きなプログラムやデータが扱えます。

そこで、プログラムで使用しない命令を削除してフリーエリアを確保しようとする場合に使われるのがNEWON命令です。

この命令によってBASICをある程度カスタマイズすることができます。

NEWONで削除される命令群は10のレベルがあり0～9の番号で指定し番号が小さくなるほどレベルは高くなります。

レベルの高い番号を指定するとそれより低いレベルの命令群はすべて削除されます。

たとえば、NEWON0とするとNEWON0～NEWON9のすべての命令が削除されます。

NEWONと数字（0～9）との間にスペースを入れないでください。

なお、BASIC起動時、NEWON命令で一度削除された命令はBOOT命令もしくはIPLリセットで再度BASICを起動しない限り使用できません。

また、一度NEWON(n=0～9)を実行したあと、削除されたBASICのコマンド、ステートメントを実行しようとしても

Reserved feature または Error 34

というエラーメッセージが表示されて実行できません。



〈参考〉

- ① HELP キーを押しながら、ディスク BASIC を起動すると、自動的に NEWON 4 が設定されます。

画面 1

```
SHARP HuBASIC CZ-8FB02 Version1.0
Copyright (C) 1984 by SHARP/Hudson
Printer : CZ-800P
80030 Bytes free

Ok
█
```

- ② BASIC を起動後、希望する NEWON を自動的に設定し、すぐにプログラム入力ができるようにするには、" Start up . Bas " のプログラムを次のように変更します。(行番号 570、650 を変更)

```
570 IF INKEY$(0)=CHR$(8HE2) THEN 650
      ~~~~~
      ↓
570 IF INKEY$(0)><CHR$(8HE2) THEN 650
      ~~~~~

650 COLOR0:KEY0,"NEWON4"+CHR$(13)+"COL.7 .....
      ↓
      0～9の数値を入力するか、または数値を省略して、
      希望する NEWON のレベルを設定する。
```

- ③ 変更したプログラムをマスターディスクに

```
SAVE"0:Start up .Bas"
```

と入力してセーブします。

(例) NEWON (省略) に設定すると、BASIC 起動後は次の画面になります。

画面 2

```
SHARP HuBASIC CZ-8FB02 Version1.0
Copyright (C) 1984 by SHARP/Hudson
Printer : CZ-800P
74493 Bytes free

Ok
█
```

n	削除されるコマンド、ステートメント
省 略	すべてのコマンド、ステートメント、関数など使用可
9	MIRROR\$, KANJI\$, DTL, RANDOMIZE, WAIT, KEYLIST, KLIST, KBUF, CANVAS, LAYER, TVPW, CHANNEL, VOL,
8	VERIFY, LOAD?, "CAS:", CMT, CMT関数, REW, FAST, EJECT, APSS
7	"COM:", ON COM GOSUB, COM ON/OFF/STOP, POSITION, PATTERN, CIRCLE@, SCROLL, STICK, STRIG, PUSH, POP, ATTR\$, RUN"?. Sys"
6	CRT, ON KEY GOSUB, KEY ON/OFF/STOP ON TIMES\$ GOSUB, TIMES\$ ON/OFF/STOP
5	MKDIR, CHDIR, RMDIR, HDOFF, SET, NAME, FPOS
4	MOUSE, MOUSE関数, HCOPY
3	GET@, PUT@, CGPAT\$, DEVI\$, DEVO\$, LPOUT CONSOLE#, COPY
2	LIST, LLIST, DELETE, RENUM, AUTO, EDIT, TRON, TROFF, SAVE, SEARCH, KILL, CONT, SAVEM, エラーメッセージ
1	PLAY, PLAY@, MUSIC, MUSIC@, TEMPO, SOUND, SOUND@, CBLACK
0	WINDOW, LINE, PSET, PRESET, CIRCLE, POLY, PAINT, PAINT@, SYMBOL, POINT, PRW, CLICK, NEWONn, PALET, PALET@

## 6

## フリーエリアの大きさを確認するには

フリーエリアはFRE関数によって内容を確認することができます。

FRE関数は

FRE (n)

という書式で書くことができ、nの値が1のときメインメモリのフリーエリア、nの値が2のときグラフィックVRAMのフリーエリア、nの値が0のとき、それらの合計がこの関数の値になります。

# 7章

## ファイルについて

「ファイル」とは、日常、「書類などの情報をバインダーなどにとじたもの」の意味で使われていますが、BASICの「ファイル」もこれと同じ意味をもっており、違うのは、情報を記録しておくものが「バインダー」ではなく、「電気や磁気による記憶装置（デバイス）」になっている点です。

ファイルは、一つの単位として取り扱われる関連したレコード（record）から成り、そこに、プログラムやデータの集合が記録されます。

### 1

## ファイルの種類

ファイルは、アクセス<sup>1)</sup>の仕方によって、シーケンシャルアクセスファイルとランダムアクセスファイルに分類されます。

- 1) アクセス（access）メモリやI/Oを読み書きすることは、すべてアクセスといいますが、ここでは、「ファイルを読み書きする」の意味で使用しています。

### 1.1 シーケンシャルアクセスファイル

ファイル中のデータが書き込まれた順に並んでおり、データを読み書きする際には、先頭から順番にしかできないようなファイルのことを、シーケンシャルアクセスファイルといいます。

一般に、このファイルではデータが順序良く並んでいて、無駄がありませんが、特定のデータだけを読み込むことはできません。そのため、データの内容を変更する場合は、内容を変更しないデータも順に読み出し、処理を行なった後、別のファイルに書き込まなければなりません。

### 1.2 ランダムアクセスファイル

ファイル中の特定のデータの記憶位置を直接指定して読み書きできるようになっているファイルを、ランダムアクセスファイルといいます。

この場合、データは1レコード（＝256バイト）単位で読み書きされます。

このファイルでは、任意のデータを指定できるので、シーケンシャルアクセスファイルではできなかった、特定のデータの変更、追加、削除をすることができます。

### 2

## デバイス

バインダーにとじたファイルは書棚にしまって整理するように、コンピュータでも、ファイルを「デバイス」と呼ばれる書棚にしまって整理します。

デバイスには、次のようなものがあります。



- フロッピーディスク
- カセットテープ
- グラフィックメモリ
- 外部メモリ (EMM)
- ハードディスク
- ディスプレイテレビ
- キーボード
- プリンタ
- RS-232C

このうち、フロッピーディスク、グラフィックメモリ、外部メモリ、ハードディスクなどのデバイスでは、シーケンシャルアクセスファイルおよびランダムアクセスファイルのいずれも使用できますが、カセットテープなどのデバイスでは、シーケンシャルアクセスファイルしか使用できません。(詳しくは「5. デバイス名」の項を参照してください。)

## 3

## ファイルの管理

書棚にしまってあるファイルには見出しを付けておくように、デバイスにしまってあるファイルにも名前が付けられています。

さらに、ファイルを探しやすいように、名前などの情報は、ファイルとは別の場所に保管されています。フロッピーディスクなどのようにランダムアクセスファイルを扱うことのできるデバイスでは、名前などの情報をディレクトリと呼ばれる場所にまとめて書かれています。カセットテープなどのようにシーケンシャルアクセスファイルしか扱うことのできないデバイスでは、1カ所にまとめずに各ファイルの先頭に名前などの情報を記録しています。したがって、これから述べようとするディレクトリについては、カセットテープなどのデバイスには利用できませんので、注意してください。

ファイルの管理はこのディレクトリによって行なわれていますが、本機ではそれを階層ディレクトリと呼ばれる方法によって行なっています。

以下階層ディレクトリについて説明します。

現在、パーソナルコンピュータの記憶媒体として、フロッピーディスクが多く使われており、比較的安価で持ち運びに便利なため、保存するファイルの数が増えると、何枚かのフロッピーディスクに分けて記録し、それぞれにインデックスシールを貼って区別するようにします。ところが、ハードディスクのようにフロッピーディスク30枚分ものデータを扱えるデバイスでは、1台の中にすべてのファイルを保存することができます。このように、何枚ものフロッピーディスクに分けて行なっていたことを1つのハードディスクで行なおうとすると、ファイルの管理上問題が生じてきます。

たとえば、ディスクに入っているファイルのリストを見るためにFILESコマンドを実行するとします。このとき、ファイルの数が10や20であれば、その中から目的のファイルを見つけ出すことはそれほど苦になりませんが、ファイルの数が100、200となった場合は、考えただけでも探す意欲がなくなってしまいます。

そこで考え出されたのが階層ディレクトリと呼ばれるファイルの管理方法です。

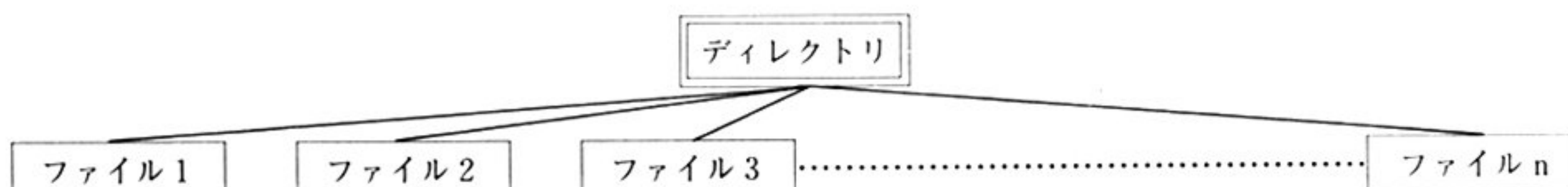
### 3.1 ディレクトリ

ディレクトリというのは、ディスクの中に記憶されているファイルの登録簿のことで、ちょうど本の目次に当たります。すなわち、ディスクにおいて、ファイル名やその種類、属性、ディスク上の位置、サイズ、作成年月日などファイルに関する情報が書かれている部分をいいます。

FILES コマンドを実行すると、このディレクトリを参照して、その中に含まれているファイル名の一覧表が表示されます。

### 3.2 単層ディレクトリ

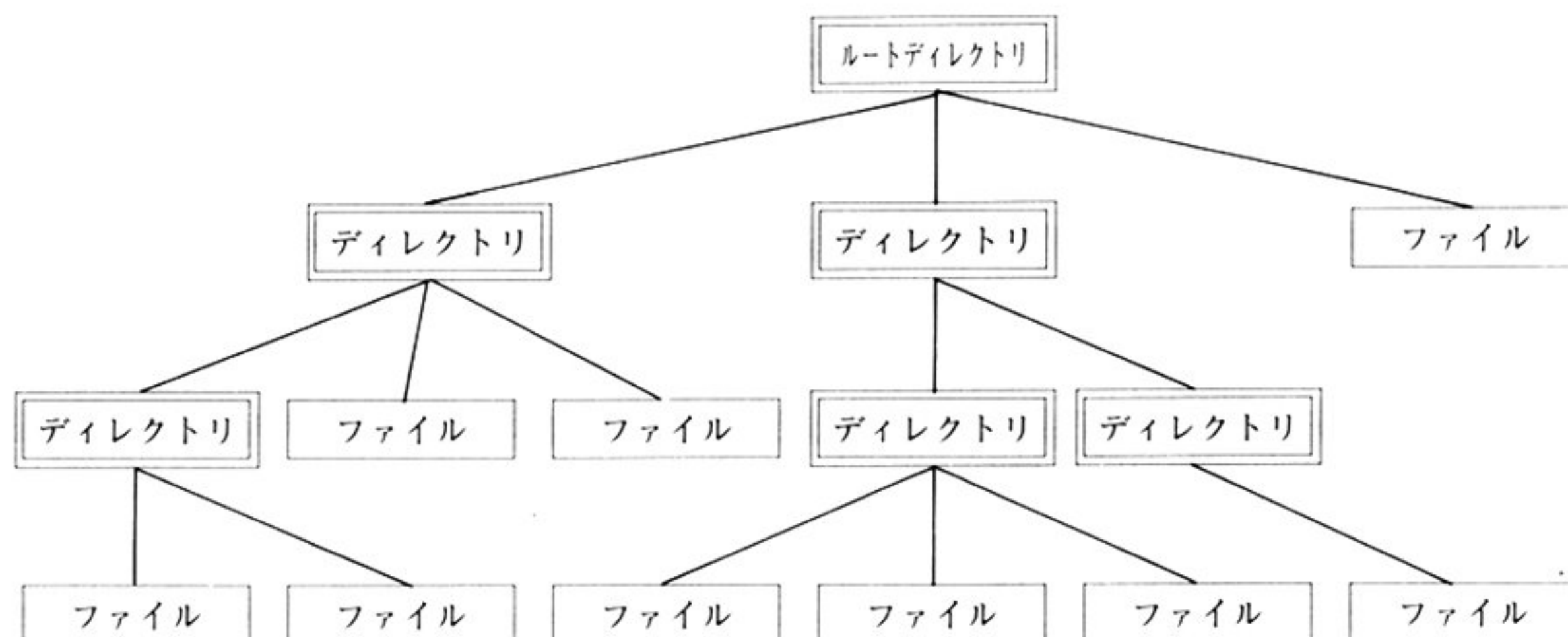
従来のファイルの管理は、ディスク1枚に1つのディレクトリ、すなわち、下図のような単層で行なわれていました。



このとき、FILES を実行するとファイル1、ファイル2、……、ファイルn のすべてのファイル名が一度にリストされることになります。

### 3.3 階層ディレクトリ

容量の大きなデバイス上のファイルを効率的に管理するために考え出されたものが、下図のような木構造をもつ階層ディレクトリです。

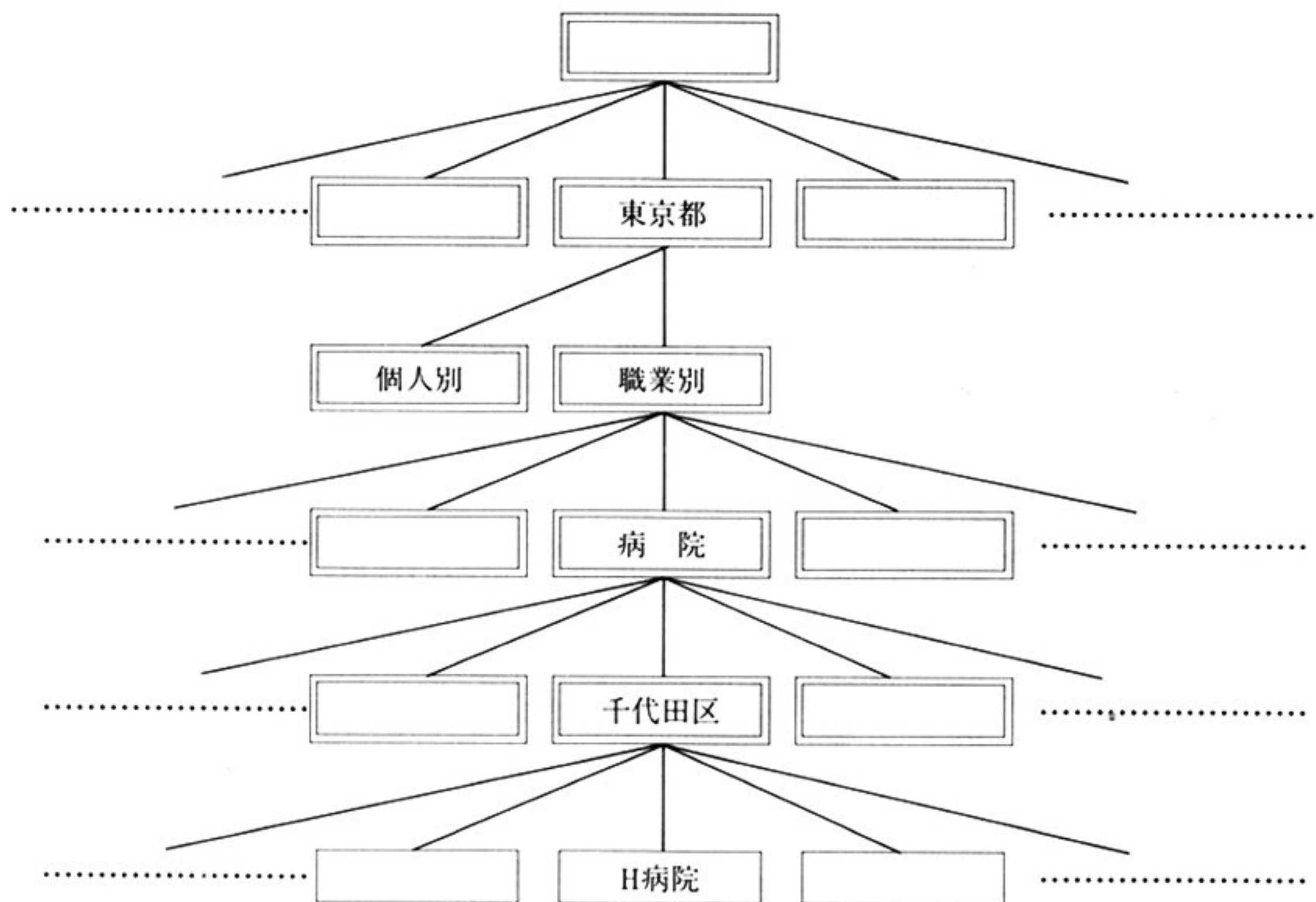


たとえば、国内のHという病院の電話番号を電話帳を使って調べる場合を考えます。H病院が東京都千代田区にあったとすると、まず東京都23区の電話帳が必要となります。電話帳は個人別のものと職業別のものとに分かれていますが、病院を調べる場合は職業別のものを使います。次に電話帳の目次から病院の項のページを調べ、そのページを開きます。すると、病院名と住所、電話番号が列記してありますが、各区ごとに項目が分かれています。そこで今度は千代田区の項目を探し出し、さらにその項のところに列記してある病院の中からH病院を探し出し、電話番号を知ることができます。つまり、H病院の電話番号を見つけるまでの経路を示すと、

電話帳の山→東京都→職業別→病院→千代田区→H病院  
となります。

この例を、階層ディレクトリに当てはめると、東京都、職業別、病院、千代田区がディレクトリに対応し、H病院というのがファイルに対応します。





本機の階層ディレクトリでは階層の経路を表わすのにスラッシュ（／）を使用します。  
したがって前記の例は次のように表わされます。

／東京都／職業別／病院／千代田区／


このときファイルにあたるH病院にたどり着くまでの経路をパスといい、ディレクトリを／で区切った

／東京都／職業別／病院／千代田区／

をパス名といいます。

また階層ディレクトリの本構造の最上部のディレクトリのことをルートディレクトリと呼び、先頭の"／"がルートディレクトリを指しています。

### 3.4 ディレクトリの作成方法

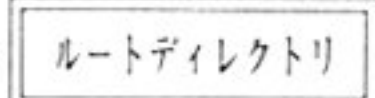
ディスクのフォーマットを行なうとディレクトリが初期化されファイルはすべて消去されます。  
この状態でF I L E S コマンドを実行すると、(ディスクドライブ1ならF I L E S " 1 : " )

×× C l u s t e r s   f r e e

P a t h   n a m e   " × : / "

と表示されます。

この表示はディスク上にルートディレクトリのみが存在していることを示しています。

★  ルートディレクトリ

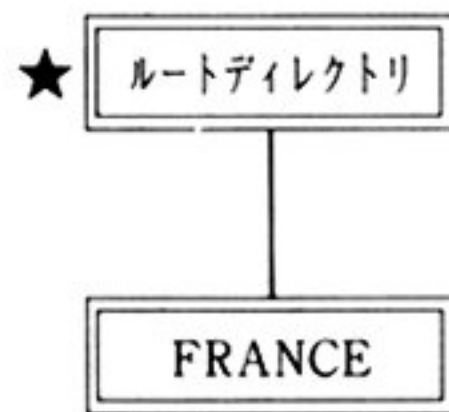
ここに新しくディレクトリを作成する場合、M K D I R というコマンドを使用します。

たとえば、ルートディレクトリの下にF R A N C E というディレクトリを作成する場合、

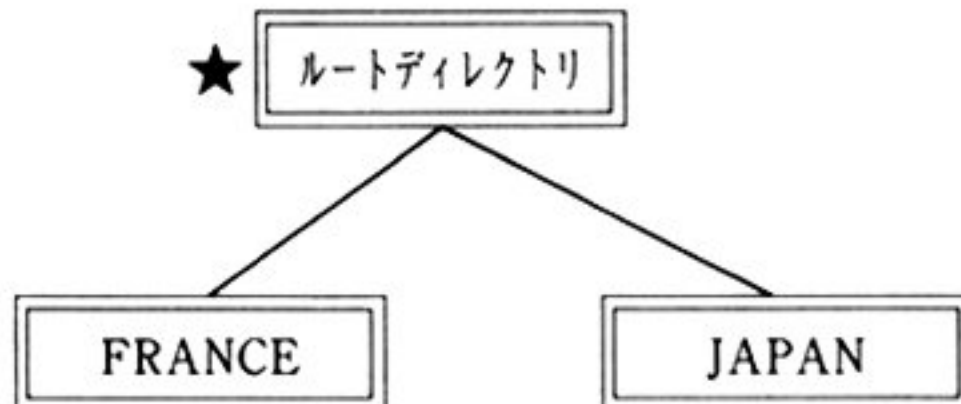
M K D I R " F R A N C E " 

とします。





さらに、JAPANというディレクトリを追加する場合は、  
**MKDIR "JAPAN"**   
 を実行します。



もちろん、ここでプログラムファイルやデータファイル（「プログラム、データファイルの保存と再生」参照）を作成することもできます。

たとえば、適当なプログラムを書いてfile 1というファイル名で記録する場合

**SAVE "file 1"** 

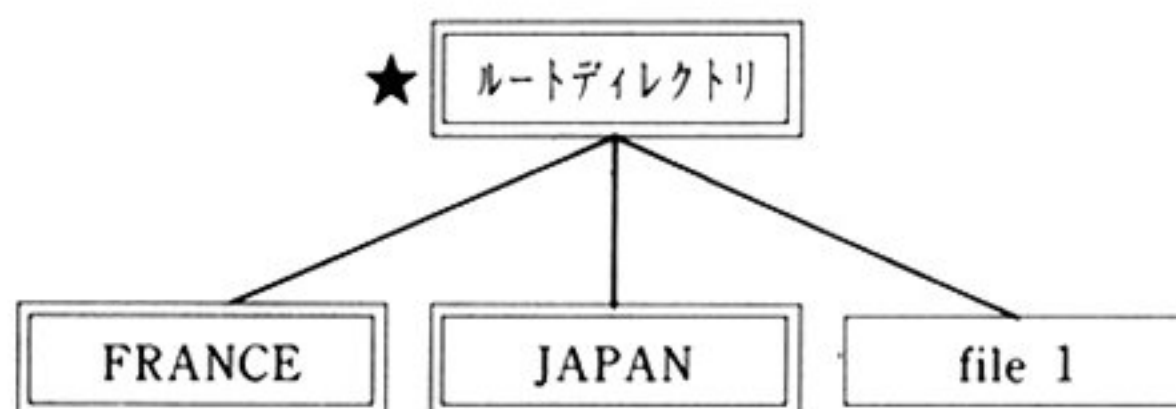
を実行します。

FILESコマンドを実行すると、

```

XX Clusters free
Path name    "X:/ "
Dir*         "X:FRANCE    .DIR"
Dir*         "X:JAPAN     .DIR"
Bas          "X:file 1    ."
  
```

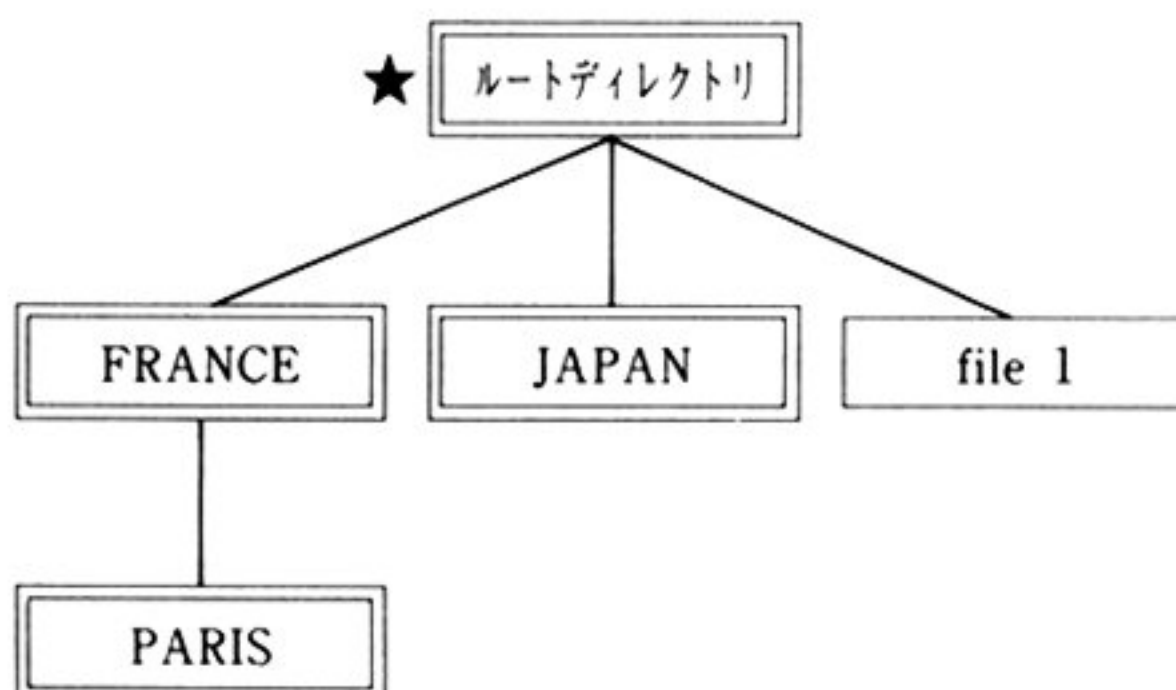
と表示されます。



FRANCEというディレクトリの下にPARISというディレクトリを作るには、

**MKDIR "FRANCE/PARIS"** 

を実行します。



さらに下の階層を作る場合も同様に " / " で区切って追加します。

```
MKDIR "...../...../...../....."
```

### 3.5 カレントディレクトリの変更

階層ディレクトリ構造では各ディレクトリはそれぞれ独立してファイルのアクセスを行ないます。したがって、同じファイル名のファイルを別々のディレクトリの中に作成することができます。

ファイルをアクセスする場合、そのファイルの存在するディレクトリに移動します。その後はそのディレクトリを起点としてファイルを指定することができます。このディレクトリのことをカレントディレクトリと呼びます。カレントディレクトリの移動には **CHDIR** コマンドを使います。

たとえば、カレントディレクトリをルートディレクトリから FRANCE というディレクトリに移す場合、

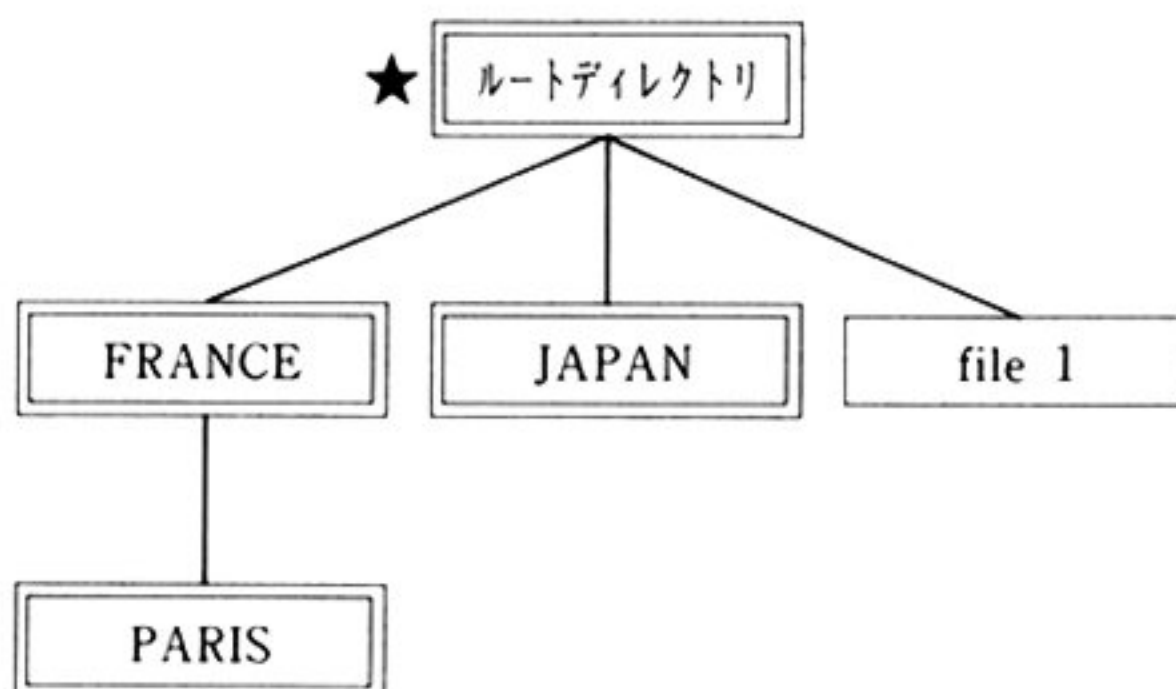
```
CHDIR "FRANCE"
```

を実行します。

このとき FILES コマンドを実行すると、

```
XX Clusters free
Path name      "X:/FRANCE/"
Dir*           "X:PARIS      .DIR"
```

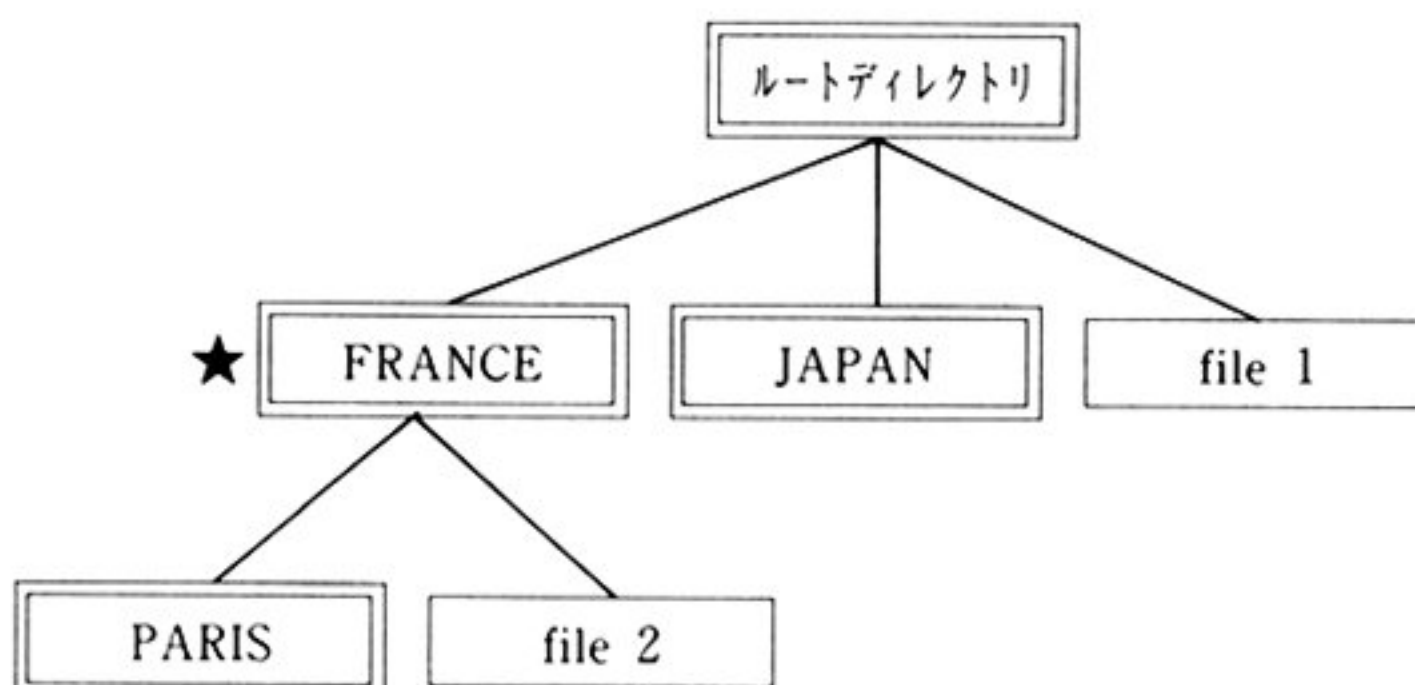
と表示されます。



ここで、プログラムを file2 というファイル名で記録する場合、

```
SAVE "file2" ↵
```

とします。



ここからさらにその下の PARIS というディレクトリに移るには、

```
CHDIR "PARIS" ↵
```

を実行します。



またカレントディレクトリをルートディレクトリに戻すには、

**CHDIR "/"** 

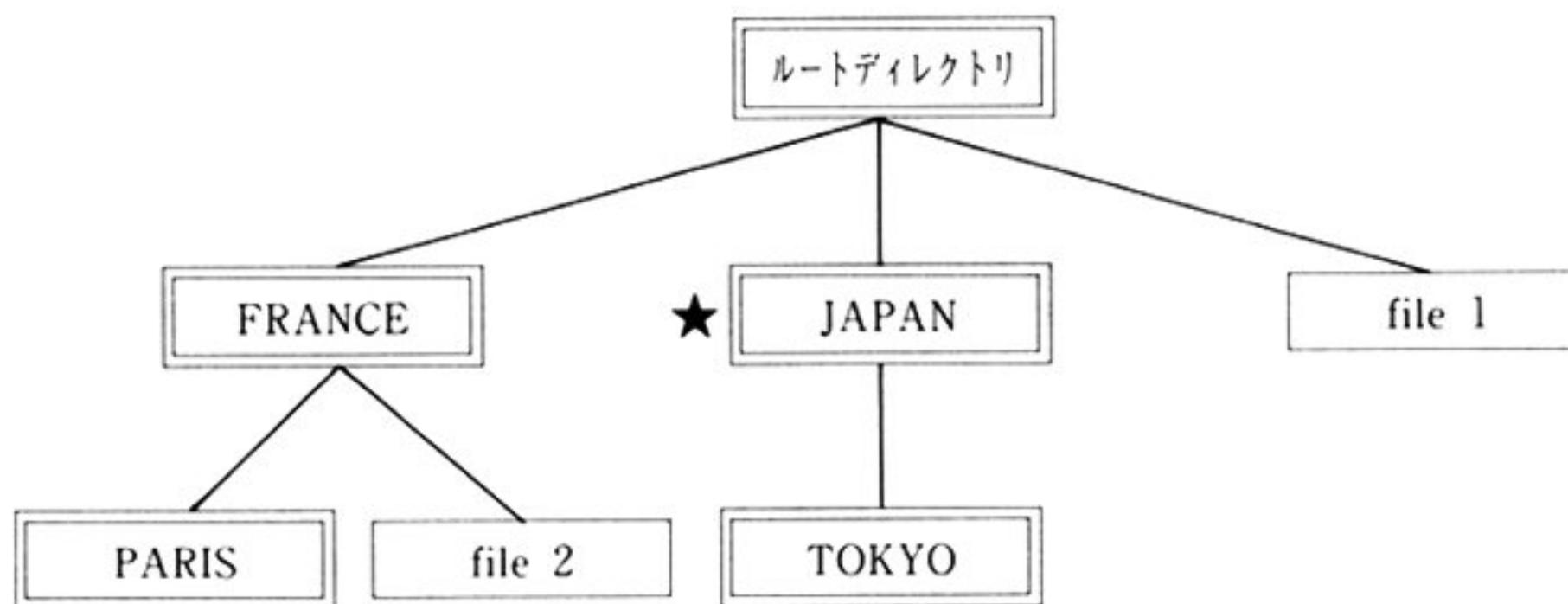
とします。

JAPANというディレクトリの下にTOKYOというディレクトリを作るには、さきほどの方法で、

**MKDIR "JAPAN/TOKYO"**

としてもよいのですが、いったんカレントディレクトリをJAPANに移してから作る方法もあります。

**CHDIR "JAPAN"**   
**MKDIR "TOKYO"** 



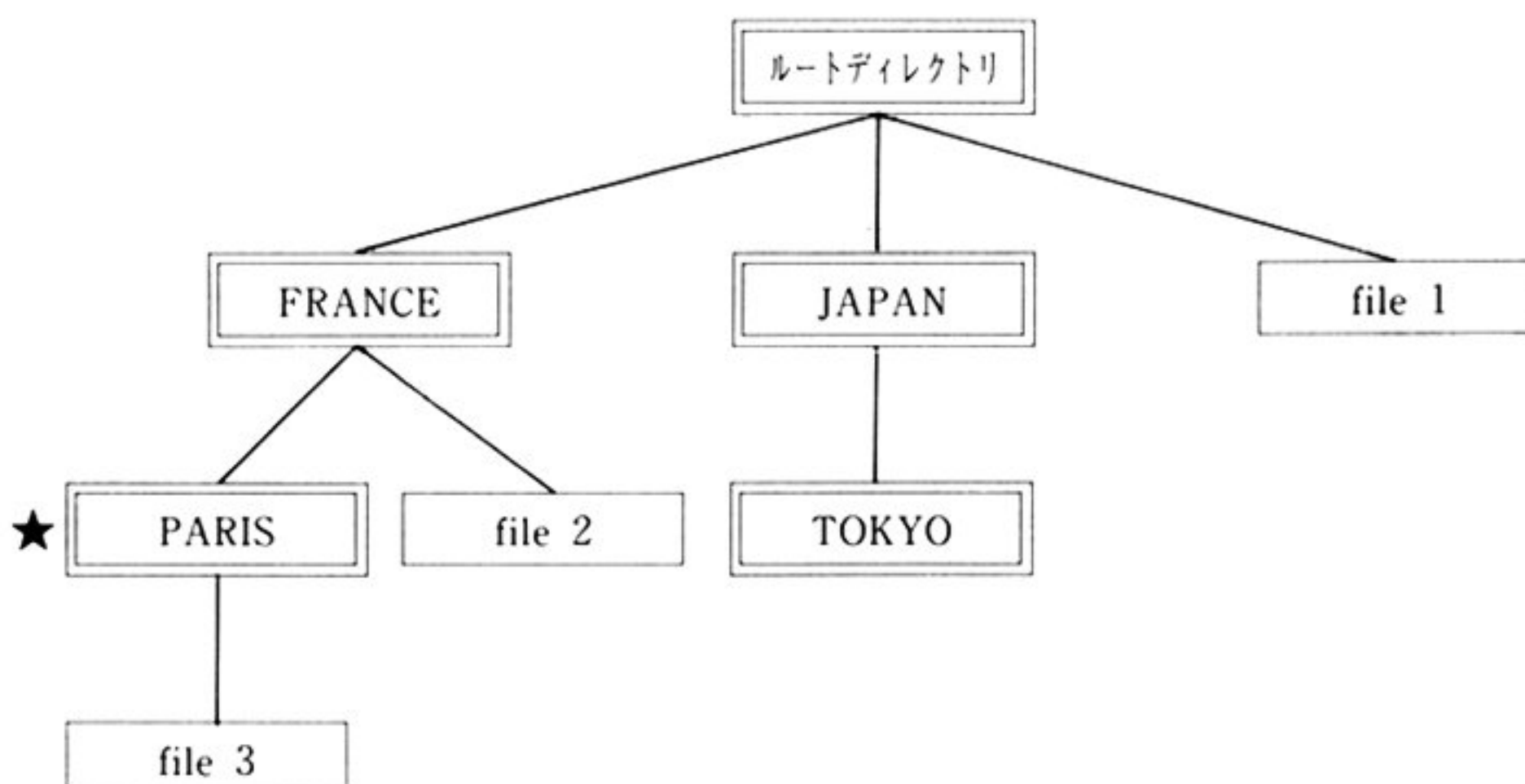
この状態でPARISというディレクトリの下にファイルを作りたいときは、

**CHDIR "/FRANCE/PARIS"** 

を実行して、カレントディレクトリをPARISに移してから、プログラムを作り、

**SAVE "file3"** 

とします。



ここでFILESを実行すると、

×× Clusters free

Path name "×:/FRANCE/PARIS"

Bas "×:file3"

と表示されます。

### 3.6 ディレクトリの削除

不要なディレクトリを削除するには **RMDIR** というコマンドを実行します。

今、カレントディレクトリを **JAPAN** として、その下の **TOKYO** というディレクトリを削除する場合、

```
CHDIR "JAPAN"
RMDIR "TOKYO"
```

とします。

このときディレクトリ **"TOKYO"** にあるファイルおよびディレクトリはあらかじめ消去されている必要があります。

もしファイルが消去されていないければ **KILL** コマンドで消去してください。

## 4

## ファイルの指定

**BASIC** からファイルをアクセスするときは、シーケンシャルアクセスファイル、ランダムアクセスファイルのいずれの場合も、まず「書棚の所まで行なって、使用したいファイルを取り出し、机の上に運んで開く」という操作をしなければなりません。このことを「ファイルをオープンする」といいます。

コンピュータには、デバイスと呼ばれる何種類かの書棚があり、いくつかのファイルを同時に使用できるように、机が何台か用意されています。

ファイルをオープンするには、書棚と使用したいファイルと机を決定する必要があります。書棚、使用したいファイル、および机を決定することを、**ファイルの指定**といいます。

ファイルの指定は、どの書棚かを示すデバイス名、分類を示すパス名、使用したいファイルを示すファイル名、および机を示すファイル番号によって行ないます。

ファイルの指定は、**ファイルディスクリプタ**と呼ばれる指定子によって行ないます。ファイルディスクリプタは、次の書式で書かれます。

" [デバイス名:]    [パス名]    [ファイル名] "

デバイス名: ..... 0 : ~ 3 : 、 F 0 : ~ F 3 : 、 H D 0 : ~ H D 3 : 、 C A S :

MEM 0 : 、 M E M 1 : 、 E M M 0 : ~ E M M 9 : 、 C R T : 、 S C R :

K E Y : 、 L P T : 、 C O M :

パス名..... [ / ] ディレクトリ [ ; パスワード ] / ディレクトリ [ ; パスワード ] / ... /

ファイル名..... 名前 [ . エクステンション ] [ ; パスワード ]

(例) LOAD " 0 : / FRANCE / PARIS / file1 "

↑

デバイス名

↑

パス名

↑

ファイル名

OPEN " R " , # 1 , " H D 0 : / 営業部 ; S H 0 2 3 / P R O G 1 . T S T ; T A K O "

↑

デバイス名

↑

パス名

↑

ファイル名

注) ディレクトリの指定は、フロッピーディスクなどのように、ランダムアクセスファイルを扱えるデバイスに対してのみ有効です。

## 5

## デバイス名

デバイス名は、ファイルの存在する書棚（デバイス）を指定するもので、半角文字で最大4文字の文字列とそれに続くコロンの（:）で表わされます。

次に示すような種類があります。

デバイス名	指定されるデバイス	用 途		
		シーケンシャル入力	シーケンシャル出力	ランダム入出力
0 :   3 :	ディスク 0 (3 または 5 インチ版)   ディスク 3 (3 または 5 インチ版)			
F0 :   F3 :	ディスク 0 (8 インチ版)   ディスク 3 (8 インチ版)			
HD0 :   HD3 :	ハードディスク 0   ハードディスク 3			
CAS :	カセットテープ			
MEM0 : MEM1 :	グラフィックメモリ 0   グラフィックメモリ 1	 	 	 
EMM0 :   EMM9 :	外部メモリ 0   外部メモリ 9			
CRT : SCR :	ディスプレイテレビ   ディスプレイテレビ		 	
KEY :	キーボード			
LPT :	プリンタ			

通信デバイス

COM :	R S -232 C    ポート			
-------	-------------------	---	---	--

\* " COM : " は特にシーケンシャル入出力として使用することができます。



パス名は、アクセスしようとするファイルが木構造をもつディレクトリのうちのどこにあるかを示します。

パス名は、次の規則に従ってつけられます。

- a) パス名はスラッシュ (/) によっていくつかの階層部分に区切られています。

／ディレクトリ／ディレクトリ／……／

- b) 1つのディレクトリは、半角文字で1～13文字の文字列で表わされています。このとき、全角文字は、1文字で2文字分と数えます。

- c) 各ディレクトリの後ろにセミコロン (;) をつけて、続けてパスワードを指定することができます。パスワードはファイル内容の秘密を守るためのもので、いったん指定すると、以後パスワードを省略してファイルをアクセスすることができなくなります。

／ディレクトリ;パスワード／……／

- d) デバイス名、ディレクトリ、セミコロン (;)、パスワードを合わせて、半角文字128文字以内でなければなりません。

デバイス名:／ディレクトリ;パスワード／……／

128文字以内

- e) パス名の先頭のスラッシュ (/) はルートディレクトリを指しています。したがって、先頭のスラッシュを省略すると、現在のカレントディレクトリの下ディレクトリを指定することになります。

- f) パス名の指定の仕方が違うときは、「Bad file descriptor」のエラーが出ます。

(例) /営業/鈴木/

／営業／鈴木;AYAKO／

／技術;T007／田中;Yoshio／

注) パス名の指定はフロッピーディスクなどのように、ランダムアクセスファイルを扱えるデバイスに対してのみ有効です。

ファイル名は、使用するファイルの名前で、デバイス名で指定されたデバイス上のどのファイルかを示します。

ファイル名は、次の規則に従ってつけられます。

- a) ファイル名は3つの部分に区切ることができます。

名前[.エクステンション][;パスワード]

- b) 名前は、半角文字で1～13文字の文字列、エクステンションは1～3文字で表わされます。ファイル名は、名前、ピリオド、エクステンション、セミコロン、パスワードをあわせて31文字以内でなければなりません。

このとき、全角文字は1文字で2文字分と数えます。

名前.エクステンション;パスワード

31文字以内

- c) 名前が13文字より長かったり、エクステンションが3文字より長い場合およびファイル名が31文字を越えた場合は、「Bad file descrypter」のエラーが出ます。
- d) パスワードはファイルの機密を保持するためにつけ、いったん指定すると以後パスワードを省略してファイルをアクセスすることができなくなります。

(例) TEST. Asc

在庫管理. Bin

Telephonelist. Bas; KAMISAN

## 8

## ファイル番号

ファイル番号は、ファイルのオープン時、用意される机の番号で、0～15の整数で表わされます。使用する机（ファイル）の数は、MAXFILESステートメントで指定することができます。

(例) MAXFILES 2……1、2の2つの机（ファイル）を使用することができます。



# 8章

## プログラム、データファイルの保存と再生

本機のメインメモリにキーボードから入力したプログラムは、いったん電源を切ると、メインメモリ上には保存されません。そこで、カセットテープやフロッピーディスクなど外部ファイルへの保存が必要になります。また、プログラム中で多くのデータを処理したい場合にも、外部ファイルにデータの保存・再生が必要になります。BASICでは、このプログラム、データの保存・再生をそれぞれプログラムファイル、データファイルという形で行ないます。ここではプログラム、データの保存および再生とそれに関するコマンドを説明します。

### 1

### プログラムファイル

コンピュータのメインメモリのプログラムは、カセットテープやフロッピーディスクなどの外部ファイルにプログラムファイルとして保存・再生が可能です。その保存・再生は次のコマンドを使用して行ないます。

SAVE、LIST、SAVEM、LOAD、LOADM、VERIFY、LOAD?、  
INIT、OPTION SCREEN、KILL、DEVICE、FILES、  
LFILES、RUN、MERGE、CHAIN

## 8章

### 1.1 プログラムの保存

メインメモリ内のプログラムを外部ファイルに記録することを、プログラムの保存（セーブ）といい、SAVE、LIST、SAVEMコマンドによって実行できます。

SAVE [" [デバイス名:] ファイル名"] [, A]

デバイス名……0:~3:、CAS:、MEM0:~MEM1:、EMM0:~EMM9:  
F0:~F3:、HD0:~HD3:、COM:

メインメモリ内のBASICプログラムを指定されたデバイスへ指定されたファイル名を付けてセーブします。また、最後にAを指定するとプログラムをアスキー形式でセーブします。アスキー形式とはプログラムをキャラクタコード（アスキーコード）に変更してセーブすることです。Aを指定しないときが一般的な記録方式であり、プログラムは中間コードに変換されてセーブされます。アスキーセーブは一般的なセーブよりも多くのファイル領域が必要になりますが、後でのべるMERGEコマンドで使うことができます。

〔注意〕①ファイル名は、プログラムをセーブするときにファイルにつける名前です。名前は、英数字、カナ、記号、日本語を使用し半角文字で13文字（2バイト文字は2文字として扱います）までです。なお、名前の中に.(ピリオド)"(引用符):(コロン);(セミコロン)/(スラッシュ)は使用できません。



- ②グラフィックメモリにプログラムをセーブする場合、あらかじめ次の2つのコマンドを実行しておく必要があります。

**OPTION SCREEN n** (n: 2~3)

グラフィックメモリを外部記憶装置として使用することを定義します。その場合、nの値は2~4を指定します。詳しくは、『BASICリファレンスマニュアル』を参照してください。

**INIT "MEM { 0 } :**  
**{ 1 }**

グラフィックメモリをイニシャライズしセーブできる状態にします。ダイレクト命令で実行した場合、「Are you sure? (y or n)」と表示されますので[Y]キーを入力してください。

- ③ディスクにプログラムをセーブする場合、ディスクをフォーマットしておく必要があります。ディスクのユーティリティプログラムを参照してください。
- ④デバイス名にCOM:を指定した場合には、RS-232Cポートへのプログラムの送り出しが行なわれます。この場合には、Aオプションをつけなくてもアスキー形式でプログラムを送り出します。

では、実際にプログラムをセーブしてみましょう。次のプログラムを「X1-WORLD」という名前でセーブしてみることにします。


[例] 

```
10 INPUT "N=";N
20 FOR P=1 TO N
30 PRINT "X1-WORLD"
40 NEXT P
50 END
```

\*プログラムをRUNすると、回数をきいてきますから、数字を入力してください。入力した回数だけ「X1-WORLD」と表示します。


上のプログラムを入力後、次のようにしてセーブを行ないます。

- カセットテープにプログラムをセーブする場合

**SAVE "CAS: X1-WORLD"** 


Ok

- フロッピーディスク（ドライブナンバー1）にプログラムをセーブする場合

**SAVE "1: X1-WORLD"** 

Ok

- そのセーブをアスキー形式で行なう場合

**SAVE "1: X1-WORLD", A** 


Ok

- グラフィックメモリにセーブする場合


**OPTION SCREEN 2** 

Ok

**INIT "MEM 0 :** 

Are you sure? (y or n) Y 

Ok

**SAVE "MEM 0 : X1-WORLD"** 


Ok

**LIST** [デバイス名:ファイル名" [,)] [(開始行番号) [- (終了行番号)]]

デバイス名…SAVEコマンド指定できるもの以外に"SCR:"、"CRT:" (どちらも画面)、"LPT:" (プリンタ) があります。

このコマンドを実行することによりメインメモリ内のプログラムを指定されたデバイスへ指定されたファイル名を付けてアスキー形式でセーブすることができます。「SAVE"ファイルディスクリブタ", A」と異なる点はセーブする行番号を指定できることです。すなわち、メインメモリにあるプログラムの開始行番号から終了行番号まで (開始行番号だけを指定した場合はその行番号だけ) をセーブすることができます。このコマンドと後でのべるMERGEコマンドを合わせてもちいることによって、あるプログラムのサブルーチン (プログラムの一部でまとまった作業を行なう部分) を他のプログラムに追加することができます。

また、すべてのパラメータを省略することによってメインメモリ中のプログラムリストを画面に表示することができます。

[例] **LIST"CAS:EXP", 30-50** 

Ok

この実行により、メインメモリにあるプログラムの行番号30～50までが、アスキー形式でカセットテープにセーブされます。

[例] **LIST"0:TEST", 10-40** 

Ok

この実行によりメインメモリにあるプログラムの行番号10から40までがアスキー形式でフロッピーディスクのドライブ0にセーブされます。

**SAVEM** [" [デバイス名:] ファイル名"], 開始アドレス, 終了アドレス [, 実行開始アドレス]

\*デバイス名……SAVEコマンドと同じものが指定できます。

メインメモリ内の機械語プログラムを外部ファイルにセーブします。

[注意] 開始アドレス、終了アドレスは機械語プログラムをセーブするときにその範囲を指定するもので、メインメモリ上の番地を指定します。これはファイルにも記録され、再生のときに使われます。

実行開始アドレスは機械語プログラムを実行するときの開始場所を指定するもので、省略するとセーブの開始アドレスが実行開始アドレスになります。

[例] **SAVEM"CAS:TEST", &H8000, &H80FF, &H800E**

メインメモリ内の&H8000から&H80FFにある実行番地&H800Eのプログラムをカセットにセーブします。

## 1.2 プログラムの再生

カセットテープ、フロッピーディスクなどの外部ファイルにセーブされたプログラムをメインメモリに移すことをプログラムの再生 (ロード) といいます。ロードはLOAD、RUN、LOADMコマンドによって実行できます。



**LOAD [" [デバイス名:] ファイル名"]**

デバイス名……SAVEコマンドと同じものが指定できます。

外部ファイルに記録されているBASICプログラムをメインメモリにロードします。

[注意] ①プログラムをロードすると、それまでメインメモリにあったプログラムは消えます。またファイル名はセーブしたときと、全く同じでなければなりません。もし、指定したファイル名のファイルがない場合には「File not found」のエラーメッセージを表示します。

②デバイス名にCOM:を指定した場合にはRS-232Cポートよりプログラムを読み込みます。この場合、通信回線より送られてくるプログラムはアスキー形式でなければなりません。

[例] SAVEコマンドの所でセーブしたファイル名"X1-WORLD"のプログラムをロードします。

カセットテープからプログラムをロードする場合、まずプログラムの先頭まで巻戻し（または早送り）した後、次のコマンドを入力します。

**LOAD "CAS:X1-WORLD" ↵**

Ok

フロッピーディスク(ドライブナンバー1)からプログラムをロードする場合、

**LOAD "1:X1-WORLD" ↵**

Ok

＊なおアスキー形式でセーブされたプログラムも同様にしてロードできます。

グラフィックメモリからプログラムをロードする場合、

**LOAD "MEM0:X1-WORLD" ↵**

Ok

**LOADM [" [デバイス名:] ファイル名" [, ロード開始アドレス]]**  
**LOADM [" [デバイス名:] ファイル名" [, [ロード開始アドレス], R]**

デバイス名……SAVEコマンドと同じものが指定できます。

外部ファイルに記録されている機械語プログラムをメインメモリにロードします。最後のパラメータRを指定した場合ロード終了後プログラムを直ちに実行します。

[注意] ロード開始アドレスを指定すると、そのアドレスから機械語プログラムのロードを開始し、この開始アドレスを省略した場合、SAVEMで指定したセーブ開始アドレスからロードが始まります。

[例] **LOADM "CAS:TEST" ↵**

Ok

カセットにセーブされた"TEST"という機械語プログラムがロードされます。

**LOADM "CAS:TEST",,R ↵**

Ok

"TEST"という機械語プログラムがロードされた後、直ちに実行されます。



**RUN [" [デバイス名:] ファイル名"]**

＊デバイス名……SAVEコマンドと同じものが指定できます。

外部ファイルに記録されているBASICプログラムをメインメモリにロードした後、直ちに実行します。また、すべてのパラメータを省略した場合、メインメモリ中のプログラムを実行します。

〔例〕 **RUN "1:X1-WORLD"** 

フロッピーディスク（ドライブナンバー1）の"X1-WORLD"プログラムをロードした後、直ちに実行します。

### 1.3 プログラムのベリファイ

カセットテープにセーブしたプログラムをメインメモリにあるプログラムとくらべて、正しくセーブされたかどうかを調べることをプログラムのベリファイといいます。このときに使うのが、VERIFYまたはLOAD?です。この2つは全く同じ意味のコマンドです。

**VERIFY [" [CAS:] ファイル名"]**

**LOAD? [" [CAS:] ファイル名"]**

＊このコマンドで扱えるデバイスは"CAS:"のみです。


カセットテープにセーブしたプログラムが正しく記録されているかどうかを調べます。なお、このコマンドの実行はカセットテープにプログラムをセーブ後すぐに行なってください。

〔例〕ここではSAVEコマンドの項で示したプログラム「X1-WORLD」をカセットテープにセーブした後、ベリファイします。

①プログラムを入力します。


②カセットテープに"X1-WORLD"をセーブします。

プログラムをセーブできるカセットテープをデータレコードにセットします。テープカウンタを0にしておくか、カウンタ番号を紙に書いておけばカセットテープのどこにセーブしたかの目印になります。

**SAVE "CAS:X1-WORLD"** 

と入力してください。画面にOkが表示され、カーソルが点滅したらセーブ終了です。

③テープをセーブした位置まで巻きもどし、次のように入力します。

**VERIFY "CAS:X1-WORLD"** 

正しくセーブされていれば次のように表示されます。

Found "X1-WORLD"

Ok

正しくセーブされていなかった場合は次のように表示されます。

Tape read error

この場合、もう一度セーブしてください。

### 1.4 ファイルの削除

メインメモリにあるプログラムを消すにはNEWを使いますが、外部ファイルに記録したプログ


ラムを消すには、K I L L という命令を使います。

**K I L L " [デバイス名:] ファイル名 "**

デバイス名……0 : ~ 3 :、MEM 0 : ~ MEM 1 :、EMM 0 : ~ EMM 9 :、F 0 : ~ F 3 :、HD 0 : ~ HD 3 :

指定したデバイス内の指定したファイルを削除します。このコマンドによって削除されたプログラムはもとにもどりません。十分注意して行なってください。

〔注意〕 デバイス名に " C A S : " を指定して、カセットに記録されたプログラムを消すことはできません。

〔例〕 **K I L L " 1 : X 1 - W O R L D " **  
O k

フロッピーディスク(ドライブナンバー1)にセーブされた " X 1 - W O R L D " プログラムを削除します。

## 1.5 ファイル名の一覧表

外部ファイルにセーブされたファイル名の一覧表はF I L E S、L F I L E Sのコマンドを実行することによって与えられます。

**F I L E S [ " デバイス名 : " ]**


**L F I L E S [ " デバイス名 : " ]**

デバイス名……0 : ~ 3 :、C A S :、MEM 0 : ~ MEM 1 :、EMM 0 : ~ EMM 9 :、F 0 : ~ F 3 :、HD 0 : ~ HD 3 :

デバイス名で指定したデバイス内にあるファイル名の一覧表がF I L E Sの場合は画面に表示され、L F I L E Sの場合はプリンタに出力されます。

〔参考〕 F I L E Sを行なった後、その中に含まれるファイルをロード、削除、実行したい場合は次のようにすると簡単に行なえます。

(1) そのファイル名の初めに表示されたB a sまたはA s cの所へ、カーソルを移動する。

(2) B a sまたはA s cのかわりにL O A D、K I L L、R U Nを入力し  キーを押す。

〔例〕 **F I L E S " 1 : " **

フロッピーディスク(ドライブナンバー1)のファイルが出力されます。

\* ファイルリストの初めに表示されるB a s、A s c、B i n、D i rには次の意味があります。

B a s……B A S I Cプログラムのファイル

A s c……アスキー形式で書き込まれたファイル

B i n……機械語プログラムのファイル

D i r……ディレクトリのファイル

〔注意〕 カセットテープなどでは、ディレクトリのファイルを扱うことはできません。

また、カセットテープに対してF I L E SおよびL F I L E Sを実行するとデバイス名



が" CAS 0："と表示されますが、" CAS："と" CAS 0："とは全く同じものです。

## 1.6 デバイス名の省略

ファイルに関するコマンドやステートメントを使うプログラムにおいて、何度も同じデバイス名を指定しなければならないことがあります。そのような時DEVICEコマンドを使用してデバイス名を省略した場合の指定が行なえます。


**DEVICE "デバイス名"**

デバイス名……すべてのデバイス名が指定できます。


CRT：、SCR：、KEY：、LPT：、CAS：、0：～3：、  
F0：～F3：、HD0：～HD3：、EMM0：～EMM9：、  
MEM0：～MEM1：、COM：

FILES、SAVE、LOADなどのファイル入出力コマンド、およびLIST、RUNなどのコマンドにおいて、デバイス名を省略すると、このコマンドで指定されたデバイス名が指定されたことになります。

〔注意〕 DEVICEコマンドでデバイス名が指定されていない時にデバイス名を省略した場合、BASIC起動時のデバイス名（カセットBASIC起動時は"CAS："、ディスクBASIC起動時はBASICを起動させたドライブ）が指定されます。

〔例〕 **DEVICE "1："** 

Ok

**LOAD "X1-WORLD"** 

Ok

ロードされるプログラムは、フロッピーディスクのドライブ1にある"X1-WORLD"になります。

## 1.7 プログラムのマージ

メインメモリ上にあるプログラムと、外部ファイルに記録されたプログラムを合成して、1つのプログラムにすることをマージすると言います。これは、次のMERGEコマンドを実行することによって行なわれます。

**MERGE "デバイス名：ファイル名"**

デバイス名……SAVEコマンドと同じものが指定できます。

このコマンドが実行されると、指定したデバイスに入っているプログラムが読み込まれ、現在メモリに存在するプログラムと組み合わせさせた1つのプログラムが作成されます。

〔注意〕 双方の行番号が重複した場合は、ファイルから読み込まれた方が優先します。また、セーブされているプログラムはアスキー形式でセーブされている必要があります。アスキー形式でセーブされていないプログラムをマージしようとするとき「Bad file mode」のエラーが出ます。



[例] ①次のプログラムを入力してください。

```
10 DATA 10,20,30,40,50,60,70,80,90,100
20 INPUT N
30 FOR I=1 TO N
40 READ DA(I)
50 NEXT
```

②入力したプログラムをフロッピーディスク（ドライブナンバー 0）にアスキー形式でセーブします。


**SAVE "0:TEST", A** 

Ok

\*カセットテープを使用する場合はデバイス名を "0:" から "CAS:" に変えて実行してください。

\*LIST コマンドによっても行なうことができます。

③メインメモリのプログラムを消します。


**NEW** 

Ok


④メインメモリにマージする次のプログラムを入力してください。

```
60 FOR J=1 TO N
70 KEI=KEI+DA(J)
80 NEXT
90 PRINT KEI
```

⑤マージを実行します。

**MERGE "0:TEST"** 

⑥ここでメインメモリのプログラムリストをとります。次のようにマージされた1つのプログラムになっていることが確認できます。

```
LIST 
10 DATA 10,20,30,40,50,60,70,80,90,100
20 INPUT N
30 FOR I=1 TO N
40 READ DA(I)
50 NEXT
60 FOR J=1 TO N
70 KEI=KEI+DA(J)
80 NEXT
90 PRINT KEI
Ok
```

\*このプログラムは行番号 10～50 で DATA 文から指定数だけデータを読み込み行番号 60～90 でその合計を出し表示するというものです。

## 1.8 プログラムのチェイン

プログラムが長すぎて、メインメモリで処理できない場合があります。この場合プログラムをいくつか分割し、それぞれ別のファイル名で外部ファイルに記録しておきます。そして、1つのプログラム実行後、自動的に次のプログラムをロードして実行するようにします。このことを「プログラムをチェインする」といい次のCHAINコマンドを使用して行います。

**CHAIN " [デバイス名:] ファイル名 "**

デバイス名……SAVE コマンドと同じものが使えます。

メインメモリ内にあるプログラムの変数を保護し、指定したデバイス内の指定したファイル名のプログラムをロードして実行します。

〔注意〕 ロードする前にオープンされていたデータファイルは、このコマンドでは閉じません。

〔例〕 サンプル(1)

```
10 DIM IN(20)
20 INPUT "DATA ノ 数",N
30 FOR L=1 TO N
40 PRINT L;"ハ"ンメノ DATA
50 INPUT IN(L)
60 NEXT
70 CHAIN"0:SAMPLE2"
```

〔注意〕 カセットテープを使用するときは、行番号70を次のように変えて入力してください。

```
70 CHAIN"CAS:SAMPLE2"
```

サンプル(2)

```
10 FOR M=1 TO N-1
20 B=ABS(IN(M)-IN(M+1))
30 PRINT
40 PRINT IN(M);TAB(10);"-";IN(M+1);TAB(22);"=I";B;"I"
50 NEXT
```


- ①サンプル(2)のプログラムを入力し、フロッピーディスク（ドライブナンバー0）に" S A M P L E 2 " というファイル名でセーブします。

SAVE " 0 : S A M P L E 2 " 


O k

〔注意〕 カセットテープを使用する時は、次のように入力してセーブしてください。

- ②メインメモリをクリアします。

NEW 

O k

SAVE " CAS : S A M P L E 2 " 

O k

- ③サンプル(1)のプログラムを入力し、実行します。データの個数とデータを入力すると隣り合う2つの数の差をとって、その絶対値が表示されます。

```
RUN 
DATA ノ 数3 
1 ハ"ンメノ DATA 
? 143.3 
2 ハ"ンメノ DATA 
? 12345.67 
3 ハ"ンメノ DATA 
? .410 
```

143.3 - 12345.67 =I 12202.37 I

12345.67 - .41 =I 12345.26 I

O k

サンプル(1)の行番号10～60はデータの入力を行なうだけですが、70行のCHAINコマンドが実行されることにより、フロッピーディスク（ドライブナンバー0）またはカセットテープからサンプル(2)のプログラムがロードされ実行されます。また、サンプル(2)のプログラムはデータの処理と表示を行なうだけですが、CHAINコマンドによってサンプル(1)の変数が保護されているため、サンプル(1)のプログラムで入力したデータが処理され表示されます。ただし、CHAINコマンドが実行されるとサンプル(1)のプログラムは消去されますので注意してください。



## 1.9 ファイル名の付けかえ


1度セーブしたファイル名を次のNAMEコマンドを実行することによって変更することができます。

**NAME " [デバイス名:] 旧ファイル名 " AS " [デバイス名:] 新ファイル名 "**

デバイス名……0:~3:、MEM0:~MEM1:、EMM0:~EMM9:、  
F0:~F3:、HD0:~HD3:

旧ファイル名で指定したファイルのファイル名を新ファイル名に変更します。

[注意] 2つのデバイス名は同じものでなければなりません。また、デバイス名に"CAS:"は使用できません。

[例] **NAME " 0:TEST " AS " 0:SAMPLE "** 

Ok

フロッピーディスク（ドライブナンバー0）の"TEST"というファイルを"SAMPLE"というファイル名に変更します。

## 1.10 ファイルの属性指定

作成したファイルを誤って消去しないようにそのファイルにライトプロテクト（書き込み禁止）をかけたり、リードアフタライト（ファイルに書き込みを行なう際にデータを書き込んだ後自動的にベリファイを行なう）やシークレット（FILESコマンドよりファイル名の一覧を表示させてもシークレット指定されたファイルは表示されない）というファイルの属性を指定する場合にはSETコマンドを実行します。ただし、SETコマンドはカセットテープに対しては使用することができません。

**SET { #ファイル番号  
" [デバイス名:] ファイル名 " }, { "P"  
"R"  
"S"  
" " }**

ファイル番号……オープンで指定したファイル番号

デバイス名……0:~3:、MEM0:~MEM1:、EMM0:~EMM9:、  
F0:~F3:、HD0:~HD3:

属性を指定する文字として「P」と「R」と「S」を使用し、「P」を指定するとライトプロテクト（書き込み禁止）がかかり、「R」を指定するとリードアフタライトが実効され、「S」を指定するとシークレットが実行されます。属性を取り除くにはヌルストリング（" "）を指定します。

[例]

①フロッピーディスク（ドライブ0）内の「SAMPLE」というBASICプログラムファイルにライトプロテクトをかけてみましょう。

**SET " 0:SAMPLE ", "P "** 

Ok



これでライトプロテクトがかかりました。F I L E S コマンドによりファイル名のリストを表示させてみてください。「S A M P L E」というファイルのB a sの次に「\*」マークが付いています。このマークがライトプロテクトのかかったファイルであることを示します。

②リードアフタライトの指定は次のようにします。

リードアフタライトを指定するファイルをO P E Nコマンドで開いた後（ファイル番号を1とすると）


**SET # 1, " R "** 

を実行すると、以降このファイルにデータを書き込むと自動的にベリファイ動作を行ないます。このファイルをC L O S Eコマンドにより閉じた時点でこのリードアフタライトの属性は解除されます。

③例①でライトプロテクトをかけたファイルにさらにシークレット指定を行なってみましょう。

**SET " 0 : S A M P L E ", " S "** 

O k

これでシークレット指定されました。F I L E S " 0 : " と入力してファイル名の一覧を表示させてみてください。「S A M P L E」というファイル名はどこにも表示されていません。ただし、「S A M P L E」というファイルが削除されてしまったわけではなく、単にファイル名の表示を行わないというだけでそのファイルはディスクに残っています。

## 1.11 ファイルのパスワード

---

自分で作成したファイルを他人に使用されたくないという場合には、そのファイルにパスワード（合い言葉）を指定して、データをファイルに書き込みます。すると、このファイルを読み出す時には指定したパスワードを入力しないと読み出せません。このパスワードを指定してファイルのアクセスを行なうには、"名前 [ . エクステンション ] ; パスワード"という形式でファイル名を使用します。




〔注意〕

パスワードには . (ピリオド) " (引用符) : (コロン) ; (セミコロン) / (スラッシュ) 記号は使用できません。またファイル名全体の長さは31文字以内でなければなりません。

〔例〕

たとえば、メインメモリにあるB A S I Cプログラムをファイルネーム「T E S T」でパスワード「A B C」を付けてディスク（ドライブ0）にセーブするには、

**SAVE " 0 : T E S T ; A B C "** 

と入力します。F I L E S " 0 : " と入力してファイル名のリストを表示してみてください。指定したパスワードは表示されておらず、パスワードを指定せずにセーブしたファイルと表面上は何の変化もありません。しかし、L O A D " 0 : T E S T " と入力して読み出そうとしても「N o p a s s w o r d」とエラー表示され読み出すことができません。読み出すためにはL O A D " 0 : T E S T ; A B C " と入力します。これにより、指定したパスワードを知らない人はそのファイルを扱うことができなくなります。

プログラム中で使用するデータは、外部ファイルにデータファイルという形で再生・保存が可能です。また、データファイルにはシーケンシャルアクセスファイルとランダムアクセスファイルの2種類があります。

シーケンシャルアクセスファイルとは、連続的に書かれたデータファイルのことで、次のような特徴があります。

- 1) ファイルの中では、データは書き込まれた順序で並んでおり、データを保存・再生する際には先頭から順番にしか行なえません。
- 2) データの内容を変更する際には、内容を変更しないデータも順に読み出し、処理を行なった後、ファイルに書き出すことが必要です。
- 3) 記録したファイルの最後にデータを追加することができます。
- 4) ファイルの中のデータは順序よく並んでおり、無駄がありません。
- 5) ファイルの作成が比較的簡単に行なえます。

シーケンシャルアクセスファイルを使用してデータの保存・再生を行なう場合、次のコマンドを使用します。

---

**OPEN、PRINT#、WRITE#、INPUT#、LINPUT#、CLOSE、  
EOF、INPUT\$、DEVICE、OPTION SCREEN、INIT  
MAXFILES**

---

これに対してランダムアクセスファイルとは、1レコード(=256バイト)単位でその記録位置を指定できるデータファイルのことで、次のような特徴があります。

- 1) ファイル中のデータは、内部のインデックスによって直接その記憶装置を指定できるようになっているため、データの保存・再生はレコードごとにランダムに行なえます。
- 2) ファイル中の一部のデータだけを変更することができます。
- 3) 記録したファイルの最後にデータを追加することができます。
- 4) データが少ない場合でもロード、セーブを1レコードごとに行なうため無駄な部分があります。
- 5) ファイルの作成のために多くの設定が必要なため、シーケンシャルアクセスファイルに比べてプログラムが複雑になります。

ランダムアクセスファイルを使用してデータの保存・再生を行なう場合、次のコマンドを使用します。

---

**OPEN、CLOSE、FIELD、LSET、RSET、GET、PUT、  
DEVICE、OPTION SCREEN、INIT、MKI\$、MKS\$、  
MKD\$、CVI、CVS、CVD**

---

シーケンシャルアクセスファイルを使えば、ほとんどのデータ処理が行なえますが、ランダムファイルを自由に使いこなしてこそ、初めてコンピュータの価値が発揮できます。

〔注意〕データファイルもプログラムファイルと同様にKILL、FILES、LFILES、NAME、SET命令によりファイルの削除、ファイル名の一覧表示等が行なえます。



## 2.1 シーケンシャルアクセスファイルへのデータの保存・再生

シーケンシャルアクセスファイルへのデータの保存・再生を行なう場合、プログラムファイルのように1つのコマンドだけで行なうことはできません。シーケンシャルアクセスファイルの場合書き込み、読み出しのコマンド（PRINT#、INPUT#など）の前後でファイルの状態を設定する必要があります。ここでは、シーケンシャルアクセスファイルへのデータの保存・再生について、実際のプログラム例を用いて説明します。なお、コマンドの詳細に関しては『BASICリファレンスマニュアル』を参照してください。

次のプログラムを入力してください。

```
10 OPEN"O",#1,"1:DATA"
20 INPUT"名前は";NA$
30 INPUT"何才ですか";AGE
40 PRINT#1,NA$
50 PRINT#1,AGE
60 CLOSE#1
100 OPEN"I",#1,"1:DATA"
110 INPUT#1,NA$
120 INPUT#1,AGE
130 PRINT"名前は";NA$"です。"
140 PRINT"歳は";AGE;"才です。"
150 CLOSE#1
```

[注意] カセットテープを使用する場合は、行番号10および100を次のように書き換え、さらに行番号70を新たに入力してください。

```
10 OPEN"O",#1,"CAS:DATA"
70 APSS-1
100 OPEN"I",#1,"CAS:DATA"
```

入力がおわりましたら、プログラムのリストをとり、間違いがないかどうかを調べてください。それでは、プログラムの説明を行ないます。

```
10 OPEN"O",#1,"1:DATA"
```

シーケンシャルアクセスファイルへデータを書き込んだり、読み出したりするには、どのファイルにどのような操作を行なうのかを宣言する必要があります。その宣言を行なうのがこのOPENコマンドで、各パラメータによって次のことが設定されます。

"O"……ファイルへのデータの書き込み（OUTPUT）を設定します。

#1……OPENコマンドによって出力するファイル番号を設定します。

"1:DATA"……出力デバイスを設定します。

すなわち、行番号10の実行でフロッピーディスク（ドライブナンバー1）にデータの出力が可能になります。また、以後のファイル処理は、ファイル名ではなくファイル番号で行ないます。

＊この設定を行なうことをファイルを開く、あるいはファイルをオープンするといいます。

```
20 INPUT"名前は";NA$
30 INPUT"何才ですか";AGE
```

行番号20、30でシーケンシャルアクセスファイルに出力するデータを用意します。

```
40 PRINT#1,NA$
50 PRINT#1,AGE
```

行番号40、50では、ファイル番号1でオープンされたファイルにデータを出力します。シーケ

ンシャルアクセスファイルにデータを出力するコマンドは、このPRINT#の他にWRITE#があります。この2つのコマンドの違いについては『BASICリファレンスマニュアル』を参照してください。

```
60 CLOSE#1
```

すべてのデータを書き込んだならば、その内容を保持するためにオープンしたファイルをとじるという処理が必要です。そのためのコマンドがCLOSEで、行番号60ではファイル番号1のファイルを閉じています。このようにして閉じられたファイルは、再びオープンされない限り読み出すことも書き込むこともできなくなります。以上でシーケンシャルアクセスファイルへのデータの出力は終了です。

```
100 OPEN"1",#1,"1:DATA"
```

次に、書き込んだデータを読み出します。読み出す場合も書き込む場合と同様にOPEN文でファイルをオープンする必要があります。ただ、書き込みを指定していたパラメータ"O"を"I"に変え、読み出し(INPUT)状態に設定する必要があります。行番号100の実行によって、フロッピーディスク(ドライブナンバー1)のシーケンシャルアクセスファイル"DATA"からデータの読み出しが可能になります。

```
110 INPUT#1,NA$  
120 INPUT#1,AGE
```

行番号110、120ではファイル番号1でオープンされたファイルからデータを変数NA\$、AGEに読み込みます。シーケンシャルアクセスファイルでは、データを読み出す場合、書き込んだ順番に行なう必要があり、一部のデータを読み出す場合でも、その前に書き込まれたデータをいったん読み出さなければなりません。

```
130 PRINT"名前は";NA$;"です。"  
140 PRINT"歳は";AGE;"才です。"
```

行番号130、140では、シーケンシャルアクセスファイルから読み込んだデータを画面に表示します。

```
150 CLOSE#1
```

行番号150ではファイル番号1のファイルを閉じています。データ書き込みだけでなく、読み込みの場合でも、その実行が終わったならばファイルを閉じる必要があります。

また、カセットテープを使用する場合、行番号70を追加しました。

```
70 APSS-1
```

これは、行番号10から60まで作成したファイルの頭出しを行なうためのステートメントです。  
[注意] データの保存、再生を行なう場合、プログラムの保存、再生と同様にディスクならば、フォーマットが、グラフィックメモリならばOPTION SCREEN、INITが実行されている必要があります。

[例] では、実際の実行例を示します。

```
RUN  
名前は? パソコンテレビ  
何才ですか? 2  
名前はパソコンテレビです。  
歳は 2 才です。  
Ok  
■
```



入力する文字「パソコンテレビ」は全角文字、「2」は半角文字で入力してください。  
＊プログラムの実行前にKMODE1が実行されていない場合は漢字を表示しません。

## 2.2 シーケンシャルアクセスファイルへのデータの追加

シーケンシャルアクセスファイルでは、ファイルの任意の位置への書き込み、読み出しは行なえませんが、ファイルの一番最後にデータを追加することはできます。ただし、カセットテープではデータの追加ができません。ここでは、2.1のプログラムで作られたシーケンシャルアクセスファイルにデータを追加する例を用いて説明します。

次のプログラムを入力してください。

＊なお、このプログラムは2.1のプログラムの一部を変更することで簡単に作れます。

```
10 OPEN"A",#1,"1:DATA"
20 INPUT"名前は";N$
25 IF N$="END"THEN 60
30 INPUT"何才ですか";AGE
40 PRINT#1,N$
50 PRINT#1,AGE
55 GOTO 20
60 CLOSE#1
100 OPEN"I",#1,"1:DATA"
105 IF EOF(1) THEN 150
110 INPUT#1,N$
120 INPUT#1,AGE
130 PRINT"名前は";N$"です。"
140 PRINT"歳は";AGE;"才です。"
145 GOTO 105
150 CLOSE#1
```

それでは、プログラムの説明を行ないます。

```
10 OPEN"A",#1,"1:DATA"
```

データの追加でファイルをオープンする場合、パラメータを"A"にしデータの追加（APPEND）状態にしなければなりません。行番号10ではフロッピーディスク（ドライブナンバー1）の"DATA"というシーケンシャルアクセスファイルにデータを追加することを宣言しています。

```
20 INPUT"名前は";N$
25 IF N$="END"THEN 60
30 INPUT"何才ですか";AGE
40 PRINT#1,N$
50 PRINT#1,AGE
55 GOTO 20
60 CLOSE#1
```

行番号25のIF文と行番号55のGOTO文で何回もデータが入力できるようになっています。シーケンシャルアクセスファイルにデータを追加する場合OPENコマンドのパラメータ以外は、入力と同じ手続きで行ないます。追加されるデータは、それ以前のデータの後に入力順に追加されます。行番号60まででデータの追加は終了です。

```
100 OPEN"I",#1,"1:DATA"
105 IF EOF(1) THEN 150
110 INPUT#1,N$
120 INPUT#1,AGE
130 PRINT"名前は";N$"です。"
140 PRINT"歳は";AGE;"才です。"
145 GOTO 105
150 CLOSE#1
```

行番号100～150では、追加されたシーケンシャルアクセスファイルからデータの入力を行ない画面に表示します。行番号105のIF文はシーケンシャルアクセスファイルの終りをEOF

(END OF FILEの略) 関数により検出し、終わりであれば行番号150へジャンプすることを意味します。EOF関数は、最後のデータならば真の値(-1)を与え、そうでないならば偽の値(0)を与えます(詳しくは、『BASICリファレンスマニュアル』を参照してください。)  
 [例] 以下は、プログラムの実行例です。

```

RUN
名前は? 小沢
何才ですか? 28
名前は? 相原
何才ですか? 24
名前は? END
名前はパソコンテレビです。
歳は 2 才です。
名前是小沢です。
歳は 28 才です。
名前相原です。
歳は 24 才です。
Ok

```

※「名前は?」の所でENDをキー入力することによってデータの追加が終わります。

## 2.3 ランダムアクセスファイルへのデータの保存・再生

ランダムアクセスファイルもシーケンシャルアクセスファイルと同様に書き込み、読み出しのコマンド前後でファイルの状態を設定する必要があります。また、ランダムファイルは、データの入出力をファイルバッファ(メインメモリ上でデータを一時的に蓄える部分)を介して行ない、そのファイルバッファへの入出力は文字型のデータでなければなりません。そのためシーケンシャルアクセスファイルよりも次のように多くの手続きを必要とします。

### 1. データの保存

- ①ファイルをオープンします。…………… [OPEN]
- ②ランダムアクセスファイルのデータを用意します。データが数値型ならば文字型に変更を行なう必要があります。… [MKI\$, MKS\$, MKD\$]
- ③ファイルバッファに文字変数を割りつけます。…………… [FIELD]
- ④ファイルバッファにデータを書き込みます。…………… [LSET, RSET]
- ⑤ファイルバッファの内容をディスク、グラフィックメモリに書き込みます。…………… [PUT]
- ⑥ファイルを閉じます。…………… [CLOSE]

### 2. データの再生

- ①ファイルをオープンします。…………… [OPEN]
- ②ファイルバッファに文字2変数を割りつけます。…………… [FIELD]
- ③ランダムアクセスファイル上のデータをファイルバッファに読み込みます。…………… [GET]



④ファイルバッファの上のデータを取り出して処理します。

文字型に変換していた数値をもとの数値型に変換しなお

します。…………… [C V I、C V S、C V D]

⑤ファイルを閉じます。…………… [C L O S E]

[注意] カセットテープでは、ランダムアクセスファイルを扱うことはできません。

それでは、実際のプログラム例を用いて説明していきます。なお、各コマンドの詳細に関しては『B A S I Cリファレンスマニュアル』を参照してください。

次のプログラムを入力してください。

プログラム(1)……………データ保存例

```
10 OPEN"R",#1,"1:RDATA"  
20 INPUT"名前は";NA$  
30 INPUT"何才ですか";AGE$  
40 T$=MKI$(AGE$)  
50 FIELD#1,10 AS A$,2 AS B$  
60 LSET A$=NA$  
70 LSET B$=T$  
80 PUT#1,1  
90 CLOSE#1
```

それでは、プログラムの説明を行ないます。

```
10 OPEN"R",#1,"1:RDATA"
```

ランダムアクセスファイルをオープンする場合 " R " のモード ( R a n d a m f i l e m o d e ) 指定を行ないます。なお、ランダムファイルの場合入力、出力で別のモード指定を行なう必要はなく1つのO P E N文でデータの入出力をすることも可能です。行番号10では、フロッピーディスク (ドライブナンバー1) に " R D A T A " という名のランダムアクセスファイルをオープンしています。

```
20 INPUT"名前は";NA$  
30 INPUT"何才ですか";AGE$  
40 T$=MKI$(AGE$)
```

ランダムアクセスファイルに出力するデータは文字型のものだけで、数値型のものは使用することができません。そのため、数式を文字列に変換した後、出力する必要があります。数式を文字列に変換するコマンドは、行番号40のM K I \$の他にM K S \$、M K D \$があります。それぞれのコマンドに関しては『B A S Cリファレンスマニュアル』を参照してください。

```
50 FIELD#1,10 AS A$,2 AS B$
```

ランダムアクセスファイルにデータの入出力を行なう場合、1レコード (256バイト) のデータをファイルバッファに蓄えておき、1レコード分ずつ入出力を行ないます。また、そのファイルバッファは、入出力の前にデータの割り付けがなされていなければなりません。それを行なうのがF I E L D文で行番号50の場合、ファイル番号1用のファイルバッファ (256バイト) の最初の10バイトをA \$、次の2バイトをB \$に割り付けています。なおこの場合、残りの244バイトは未使用となり、たいへんぜいたくな使い方といえます。

\*F I E L D文はファイルバッファに変数の割り付けを行うだけで、データをファイルバッファおよびファイルに入出力しません。

```
60 LSET A$=NA$
70 LSET B$=T$
```

行番号60、70ではFIELD文で割り付けたファイルバッファにデータを出力しています。ファイルバッファにデータを出力する場合、このLSETの他にRSETというコマンドがあります。2つの違いについては『BASICリファレンスマニュアル』を参照してください。

```
80 PUT#1,1
```

PUT文によってファイルバッファの内容(256バイト)がランダムアクセスファイルに出力されます。行番号80では、ファイル番号1のファイルバッファにあるデータをOPEN文で指定した"RDATA"というファイルのレコード番号1に書き込んでいます。ランダムアクセスファイルには、レコード番号が記録されるようになっており、このレコード番号の指定を変えることによりデータのランダムな入出力が可能となります。

```
90 CLOSE#1
```

行番号90では、オープンされていたランダムアクセスファイルを閉じています。ランダムアクセスファイルの場合、入出力別にOPEN文を実行する必要があるため、ファイルを閉じなければ、データの入出力を1つのOPEN文、FIELD文によって行なうことができます。

[例] 実行例を以下に示します。

```
RUN
名前は何? パソコンテレビ
何才ですか? 2
Ok
■
```

※入力する文字「パソコンテレビ」は全角文字で入力してください。  
次にランダムアクセスファイルからのデータ入力について説明します。  
次のプログラムを入力してください。  
プログラム(2).....データ再生例

```
10 OPEN"R",#1,"1:RDATA"
20 FIELD#1,10 AS A$,2 AS B$
30 GET#1,1
40 AGE=CVI(B$)
50 PRINT "名前は";A$;"です。"
60 PRINT "歳は";AGE;"才です。"
70 CLOSE#1
```

それでは、プログラムの説明を行ないます。

```
10 OPEN"R",#1,"1:RDATA"
20 FIELD#1,10 AS A$,2 AS B$
```

行番号10でランダムアクセスファイルをデータ入力のためにオープンしています。ランダムアクセスファイルの場合、前に述べたようにデータの入出力でオープン方法は変わりません。行番号20では、ファイルバッファに変数を割り付けています。この場合、出力時と変数名はかえることができますが、ファイルバッファの割り付けは同じにする必要があります。

```
30 GET#1,1
```



ファイルから1レコード(256バイト)分のデータをファイルバッファに入力するにはGET文を使います。行番号30の場合、ファイル番号1のファイルバッファにOPEN文で指定した"RDATA"というファイルのレコード番号1からデータを256バイト読み込んでいます。

```
40 AGE=CVI(B$)
50 PRINT "名前は";A$;"です。"
60 PRINT "歳は";AGE;"才です。"
70 CLOSE#1
```

ファイルバッファに読み込まれたデータは、FIELD文で割り付けた文字変数を使用することによって取り出せます。また、一旦文字式に変換した数値は、CVI、CVS、CVDコマンドを実行することによりもとの数値に変換することができます。3つのコマンドの違いについては『BASICリファレンスマニュアル』を参照してください。以上のことは行番号40～60で行なわれており、行番号70では入力のためにオープンしたファイルを閉じています。

[例] それでは実行例を示します。なおこのプログラムの実行前にプログラム(1)が実行されている必要があります。

```
RUN
名前がパソコンテです。
歳は 2 才です。
Ok
■
```

\*名前の入力をする際、全角文字でパソコンテレビと入力しましたが、FIELD文でそのファイルバッファの領域を10バイトしか設けなかったため、このプログラムでは"パソコンテ"としか表示されません。データをファイルバッファに割り付ける場合、その大きさに注意する必要があります。

## 2.4 ランダムアクセスファイルに関するデータの追加・変更

ランダムアクセスファイルはシーケンシャルアクセスファイルに比べて多くの手続きを必要としますが、データの追加・変更は自由に行なうことができます。すなわち、レコード番号を指定することによって、どの場所のデータもアクセス(呼び出し)可能です。ここでは、2.3で作成したランダムアクセスファイルにデータを追加・変換するプログラム例を説明します。

次のプログラムを入力してください。

プログラム(3)……………データ追加・変換プログラム

```
10 OPEN"R",#1,"1:RDATA"
20 INPUT"名前は";N$
25 IF N$="END"THEN 90
30 INPUT"何才ですか";AGE%
40 T$=MKI$(AGE%)
50 FIELD#1,10 AS A$,2 AS B$
60 LSET A$=N$
70 LSET B$=T$
75 INPUT"レコード番号";N
80 PUT#1,N
85 GOTO 20
90 CLOSE#1
```

ほとんどがプログラム(1)の場合と同じです。行番号75で入力するレコード番号を変えることにより、どの場所のデータでも追加変更できます。なお、このプログラムでは名前に「END」を入力するまで何回でも追加・変更を繰り返すことができます。

〔例〕実行例を示します。

```

RUN
名前は何？ 小林
何才ですか？ 29
レコード番号？ 1
名前は何？ 中村
何才ですか？ 25
レコード番号？ 3
名前は何？ 早川
何才ですか？ 70
レコード番号？ 2
名前は何？ END

```

Ok

それでは、プログラム(3)で作成したランダムファイルからデータを取り出す次のプログラムを入力してください。

プログラム(4)

```

10 OPEN "R", #1, "1:RDATA"
20 FIELD #1, 10 AS A$, 2 AS B$
25 INPUT "レコード番号？"; N
28 IF N=0 THEN 70
30 GET #1, N
40 AGE=CVI(B$)
50 PRINT "名前は"; A$; "です。"
60 PRINT "歳は"; AGE; "才です。"
65 GOTO 25
70 CLOSE #1

```

この場合もまた、プログラム(2)とほとんど同じです。行番号25でアクセスするレコード番号を入力し、行番号65によって繰り返しが行なえるようになっています。

〔例〕実行例を示します。

```

RUN
レコード番号？ 1
名前は何？ 小林
何才ですか？ 29
レコード番号？ 2
名前は何？ 早川
何才ですか？ 70
レコード番号？ 3
名前は何？ 中村
何才ですか？ 25
レコード番号？ 0
Ok

```



＊ファイル番号に①を入力することによってプログラムの実行が終了します。

また、「RDATA」というファイルのレコード番号1には、プログラム(1)の実行により「パソコンテ」というデータが入っていましたが、プログラム(3)の実行例で示すようにレコード番号1に新しく「小林」というデータを書き込んだため、前のデータは消去されています。

# 9章

## ミュージック

本機には、プログラムサウンドゼネレータ（P S G）という、擬似音発生用 L S I が内蔵されています。この P S G を使うことにより音楽演奏やゲームの効果音を作ることができます。ここではこの P S G をコントロールするサウンド制御ステートメントの使い方を説明します。

なお、サウンド制御ステートメントには次のようなものがあります。

**BEEP, MUSIC, PLAY, TEMPO, SOUND, SOUND@, MUSIC@, PLAY@**

1

### ブザー音を出す

ブザー音を出すためには BEEP ステートメントを次のように使用します。

**BEEP** { 0 .....ブザー音を止める。  
1 .....ブザー音を出し続ける。  
省略.....短い「ポッ」という音を出す。 }

[例] **BEEP 1**   
O k  
**BEEP 0**   
O k

2

### 楽譜を演奏する

MUSIC、PLAY、TEMPOのステートメントを使用して楽譜の演奏（8オクターブ、3重和音）が行なえます。

**MUSIC 文字式**

文字式で指定された楽譜に従って演奏します。文字式では、音程、長さ、休符、オクターブ、音量、和音の指定を以下のようにして行なうことができます。



(1)音程.....音程は下記のようにC～Bの文字を使用して指定します。

音程	ド	ド# (レb)	レ	レ# (ミb)	ミ	ファ	ファ# (ソb)	ソ	ソ# (ラb)	ラ	ラ# (シb)	シ
指定文字	C	#C	D	#D	E	F	#F	G	#G	A	#A	B

(2)長さ.....音程の後ろに0～9の整数をつけて、音の長さを指定します。

＊音の長さは4分音符（整数5）を1としたときの相対的な値です。

また、整数の指定がない場合は前の音と同じ長さを意味します。

音の長さ	対応する整数
$\frac{1}{32}$ (32分音符)	0
$\frac{1}{16}$ (16分音符)	1
$\frac{3}{32}$ (付点16分音符)	2
$\frac{1}{8}$ (8分音符)	3
$\frac{3}{16}$ (付点8分音符)	4
1 (4分音符)	5
$1\frac{1}{2}$ (付点4分音符)	6
2 (2分音符)	7
3 (付点2分音符)	8
4 (全音符)	9

③休符.....R 0～R 9で、休みの長さを指定します。

＊数字の値は上記の長さに対応しています。

④オクターブ....." O 1 "～" O 8 "のようにアルファベットの" O "の後ろに1～8の整数をつけてオクターブの指定をします。また、音符の前に+をつけることによって上の音程を、-をつけることによって下の音程を指定できます。初期状態はO 4です。

⑤音量.....V 0～V 15で音量の指定をします。なお、この指定の次から音量が変化します。

⑥和音.....2重音や3重和音を演奏するには、上声部と下声部の間にチャンネルセパレータ"："（コロン）をはさみます。

"チャンネルA：チャンネルB：チャンネルC"

**TEMPO 数値 (30～7500)**

MUSIC文によって演奏される曲の速さ（テンポ）を指定します。

＊指定された数値は1分間あたりの4分音符の拍数を示します。

初期状態では120に設定されています。

**PLAY { 数値 }  
          { 文字式 }**

MUSIC文とTEMPO文の両方の機能を持ち、PLAYの後ろが数式の場合TEMPO文と同様の、文字式の場合MUSIC文と同様の機能を持ちます。

[例] オクターブを1から8まで変化させてからテンポを速くし、これを繰り返します。

```
10 'MUSIC or PLAY
20 WIDTH 40: INIT
30 CLS4
40 PLAY"V15R0:V15R0:V15R0" _____音量最大 休符は最小。
50 FOR T=120 TO 2000 STEP 100 _____テンポを変化させる。
60 TEMPO T
70 FOR O=1 TO 8 _____オクターブを1から8まで変化させる。
80 PLAY"O"+CHR$(48+O)+"CDEFGABAGFEDC:DEFGABAGFEDCB:EFGABAGFEDCBA"
90 NEXT O
100 NEXT T
```

MUSIC@、PLAY@ステートメントは音の長さをCTCでコントロールしていますので音を出しながら、グラフィックステートメント等を実行することができます。ただしプログラムの中でのみ使用可能です。

### 3 PSGのコントロール

本機で音を出すには、先に述べたPLAYステートメント等によって楽譜を文字列にして演奏する方法とSOUND、SOUND@を使用して直接PSGにデータを送り音を出す方法があります。ここでは、SOUND、SOUND@の使い方を説明します。

**SOUND** PSGのレジスタ番号, データ [, データ……]

**SOUND@** PSGのレジスタ番号, データ [, データ……]

SOUND、SOUND@命令は、PSGの14個のレジスタへデータを書き込んで、ゲームの効果音など様々な音を作り出す事ができます。

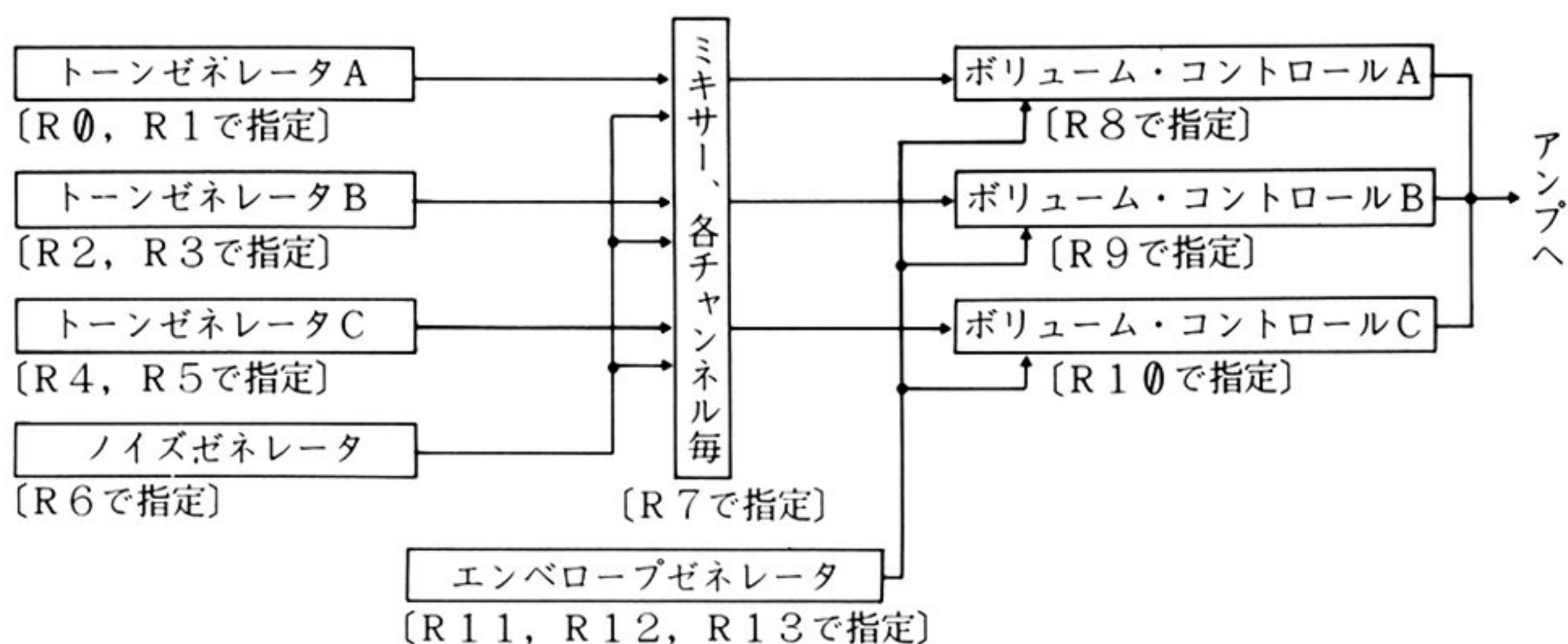
なお、SOUND命令は、1つのデータで1つのレジスタ（8ビット）を定義しますが、SOUND@命令は、1つのデータで連続したレジスタ（16ビット）を定義できます。後で説明する周波数の設定のように連続したレジスタを定義する場合にSOUND@を使用します。

また、データをカンマ（,）で区切って並べて書くと、連続した複数のレジスタにそのデータを書き込むことができます。

本機に内蔵しているPSGは、3つのトーンゼネレータ、ノイズゼネレータ、エンベロープゼネレータ、ミキサー、3つのボリュームコントロールから構成されています。

そのブロック図を次に示します。





(注) 図中の R 0 ~ R 13 はレジスタ番号です

上のブロックのレジスタにデータを送ることによりさまざまな音が発生します。

その各レジスタの機能表を以下に示します。

レジスタ番号	レジスタ機能	ビット構成								設定データ値 (15進) 番号
		7	6	5	4	3	2	1	0	
0	チャンネル A 周波数	下位 8 ビット T <sub>L</sub> (微調)								0 (高) ~ 255 (低)
1						上位 4 ビット T <sub>H</sub> (粗調)				0 (高) ~ 15 (低)
2	チャンネル B 周波数	下位 8 ビット T <sub>L</sub> (微調)								0 (高) ~ 255 (低)
3						上位 4 ビット T <sub>H</sub> (粗調)				0 (高) ~ 15 (低)
4	チャンネル C 周波数	下位 8 ビット T <sub>L</sub> (微調)								0 (高) ~ 255 (低)
5						上位 4 ビット T <sub>H</sub> (粗調)				0 (高) ~ 15 (低)
6	ノイズ周波数					5 ビット N				0 (高) ~ 31 (低)
7	チャンネル選択	IN/OUT IOB IOA		ノイズ C B A			トーン C B A			0 ~ 63 (各ビット共、0 を設定すると選択される)
8	チャンネル A 音量			M		4 ビット			0 (小) ~ 16 (大)	各チャンネルとも M=0 のとき (データ値 0 ~ 15) 4 ビットの値で 0 から 15 段階に音量設定可能 M=1 のとき (データ値 16 ~ 31) 音量はエンベロープに依存し 4 ビットデータは無効になる。
9	チャンネル B 音量			M		4 ビット			0 (小) ~ 16 (大)	
10	チャンネル C 音量			M		4 ビット			0 (小) ~ 16 (大)	
11	エンベロープ周期	下位 8 ビット E <sub>L</sub> (微調)								0 (小) ~ 255 (大)
12		上位 8 ビット E <sub>H</sub> (粗調)								0 (小) ~ 255 (大)
13	エンベロープ形状					4 ビット				0 ~ 15

\*レジスタ番号 7 のビット 6、7 は、SOUND、SOUND@命令では使用しません。

それでは各ブロックの設定方法を説明します。

#### (1) トーンゼネレータ

レジスタ 0 ~ 5 では、A、B、C 各チャンネルの周波数を決定します。周波数は音の音程にあた

り、1オクターブ音が上ると周波数は倍になります。また、その間を12に分けると半音階が得られます。

下の表は、オクターブ4の各音の周波数を表わしています。

音 階	C	#C	D	#D	E
周波数 (Hz)	261.63	277.18	293.66	311.13	329.63

音 階	F	#F	G	#G	A
周波数 (Hz)	349.23	369.99	392.00	415.30	440.00

音 階	#A	B
周波数 (Hz)	466.16	493.88

\*音階はMUSICの文字列で表わしています。

PSGでは入力周波数である2MHzを内部で16分の1に分周（周波数を分割する操作）したもの、各チャンネルの12ビットデータでさらに分周して発生させます。

求めたい周波数をf [Hz]、分周値をT [Hz] とすると次の関係があります。

$$f = \frac{2 \times 10^6}{16 \times T} \dots\dots\dots \textcircled{1}$$

たとえば、1kHzの音を発生するには、①を変形した次の式に周波数を代入することにより分周値を求めることができます。

$$T = \frac{2 \times 10^6}{16 \times f} \dots\dots\dots \textcircled{2}$$

代入すると

$$T = \frac{2 \times 10^6}{16 \times 10^3} = 125$$

この分周値を各チャンネルのレジスタ（Aチャンネルの場合、レジスタ1の上位4ビットとレジスタ中の下位8ビット）に指定するところにより1kHzの音が設定できます。

Aチャンネルに1kHzを設定する場合次のようにします。

**SOUND@0, 125**

\*この場合レジスタ0には125が、レジスタ1には0が設定されます。

また、440Hz（O4A）を指定したい場合は次のようにします。

$$T = \frac{2 \times 10^6}{16 \times 440} \approx 284$$

チャンネルBに440Hz（O4A）を指定する場合、次のようにします。

**SOUND@2, 284**

\*この場合レジスタ3には1が、レジスタ2には28が送られます。



## (2) ノイズゼネレータ

レジスタ6によって雑音の平均周波数を指定します。雑音とは周波数が不規則に変化するものを指し、レジスタ6の5ビットの値によって分周値を設定します。

たとえば、5 KHzのノイズを発生するには②の式に周波数を代入し分周値を求めます。

$$T = \frac{2 \times 10^6}{16 \times 5000} = 25$$

この値をレジスタに設定する場合1つのレジスタのみを指定すればよいためSOUNDステートメントを使います。

**SOUND 6, 25**

## (3) ミキサー (チャンネル選択)

レジスタ7では、A、B、C各チャンネル毎にトーン／ノイズ／（トーン＋ノイズ）出力の有無を選択します。たとえば、

**SOUND 7, 44** または **SOUND 7, &B101100**

と設定すると、Aチャンネルからはトーンだけ、Bチャンネルからはトーン＋ノイズ、Cチャンネルからはどちらも出力されません。

## (4) ボリュームコントロール

レジスタ番号8～10は、各チャンネルの音量を決定します。ただし、Mビット（5ビット目）の値により、4ビットデータによる0～15までの一定した音量が設定できるモード（M=0）と、自動的に音量が0～15まで変化するエンベロープモード（M=1）とを選択できます。

たとえば、

**SOUND 9, 12**

と設定するとBチャンネルの音量を12に設定できます。

## (5) エンベロープゼネレータ

レジスタ11、12では、エンベロープの周波数を指定し、レジスタ13ではエンベロープの形状を指定します。

まず、周波数の設定方法を説明します。PSGでは、エンベロープの周波数は入力周波数2 MHzを256分の1に分周し、それをレジスタ11、12の16ビットの値でさらに分周しています。そのため、エンベロープ周波数を $f_E$  [Hz]、分周値をTとすると次の関係があります。

$$f_E = \frac{2 \times 10^6}{256 \times T} \dots\dots\dots \textcircled{3}$$

たとえば、 $f_E = 0.5$  [Hz]（エンベロープ周期1 /  $f_E = 2$  [秒]）を設定するには、③式を変形した次の式に周波数を代入することにより分周値を求めることができます。

$$T = \frac{2 \times 10^6}{256 \times f_E} \dots\dots\dots \textcircled{4}$$

代入すると

$$T = \frac{2 \times 10^6}{256 \times 0.5} = 15625$$






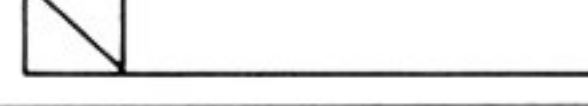
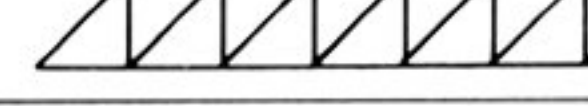
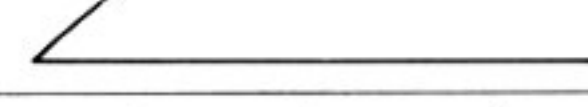
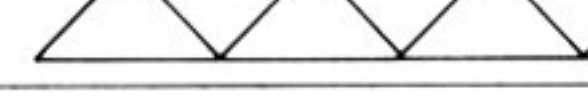
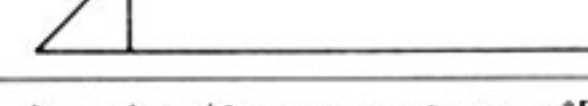
この分周値を次のように指定することによって0.5 [Hz]が設定できます。


**SOUND@11, 15625**

＊この場合、レジスタ 1 1 に 9 が、レジスタ 1 2 に 6 1 が設定されます。

次にエンベロープの形状を決定します。

それには、下の表に示すパターンに対応したデータをレジスタ 1 3 に指定することが必要です。

R 13のデータ	エンベロープパターン
0 ~ 3	
4 ~ 7	
8	
9	
10	
11	
12	
13	
14	
15	

  $1/f_E$  = エンベロープ周期

〔例〕では、実際に SOUND, SOUND@ステートメントで音を出してみます。下のプログラムを実行してください。

```

10 SOUND@0,284 ..... A チャンネルの周波数設定。
20 SOUND6,25 ..... ノイズ周波数設定。
30 SOUND@11,15625 ..... エンベロープ周期の設定。
40 SOUND13,8 ..... エンベロープパターンの設定。
50 SOUND8,15 ..... A チャンネルの音量設定。
60 SOUND7,&B111110 ..... A チャンネルの選択。
  
```

A チャンネルから 4 4 0 H z (0 4 A) の音が出ます。音をとめるには **CTRL** + **D** を押してください。次に 6 0 行目も次のように再入力してください。

```

60 SOUND7,&B110110 ..... A チャンネルのノイズとトーンを選択。
  
```

A チャンネルからは、音とノイズの両方が出力されます。次に 5 0 行目を次のように再入力してください。

```

50 SOUND8,16 ..... エンベロープモードを設定。
  
```

A チャンネルの音量にエンベロープがかかり、おもしろい音になります。

また、SOUNDステートメントでエンベロープを指定することにより、PLAY (MUSIC)ステートメントによる同一音の演奏を音符毎に区切ることができます。

```

PLAY120:PLAY"V1604C5CDDEED"
  
```

上のPLAYステートメントを、次のようにかえてみてください。ただし、文字式での音量指定 (V) は V 1 6 にします。

```

10 SOUND@11,5245:SOUND13,0:SOUND8,16
20 PLAY120:PLAY"V1604C5CDDEED"
  
```



# 10章

## 配列

コンピュータで扱うデータがプログラムの実行によって変化するような場合、**変数**というものを使います。

変数はプログラムで使われるデータの内容により**数値変数**とか**文字変数**と呼ばびます。

データの数が少ない場合には使われる変数の数は少なくてすみませんが、100個、1000個…というようにデータの数が増えると、いくつもの変数を用意しなくてはならず、プログラムが複雑になってしまいますし、メモリのムダも多くなります。

そこで、このように扱うデータ量が多くなった場合には、配列を使用すると便利です。

配列というのは一組のデータに番号を付けて規則的に並べたもので、以下に例をあげて説明します。

たとえば、郵便屋さんが、あるアパートに手紙を配達する場合を考えてみてください。そして、そのアパートが5階建てで、各階に10件ずつの家があるとしします。

さて、ここで、各家庭に手紙を配達しようとしたとき、手紙の宛名をみながら一軒一軒手紙を配っていくと、階段を何度も上ったり降りたりして、たいへん効率の悪い配り方となります。ところが、この手紙は2階の3軒目の家の手紙というように、それぞれの手紙を何階の何軒目という具合に整理してから配達すると、時間も短くて済み、効率が良く配達できます。

コンピュータでデータを扱う場合も同様で、データの数が増えるほど一組にまとめて、番号順に並べて整理をした方が、扱いやすいものとなります。

このようにデータに一連の番号を付けて、並べたものを**配列**といいます。

### 1

## 配列名と添字

配列は変数の集まりですので名前を付けて扱います。この名前を**配列名**といい、使用文字や長さは一般の変数と同様です。

配列で使用する一連の番号は( )でくくり、その配列上の位置を示しており**添字**と呼ばびます。

つまり配列データを参照する場合は、配列名と添字を指定します。

また、配列の個々のデータを**配列要素**といいます。

配列データの書き方は次のようにします。

### 配列名 (添字)

たとえばAという配列名で、5個のデータを格納する場合次のようになります。

A(5)

A(1)	A(2)	A(3)	A(4)	A(5)
------	------	------	------	------

さらに、添字はカンマ(,)で区切ってB (3, 3)のように並べることができます。  
この例では、 $3 \times 3 = 9$ 個のデータを格納できます。

B (3, 3)

B (1, 1)	B (1, 2)	B (1, 3)
B (2, 1)	B (2, 2)	B (2, 3)
B (3, 1)	B (3, 2)	B (3, 3)

B (3, 3)とした場合、添字が2個ありますので、**2次元配列**といいます。  
添字が3個ある場合には**3次元配列**、n個ある場合は**n次元配列**といいます。  
次元は一行で扱える文字数との関係から125次元程度まで作成できます。

## 2

## 配列宣言

配列を使用する場合は、あらかじめ使用するデータ領域を確保する必要があります。  
これを**配列宣言**といい**DIM**ステートメントを使います。  
**DIM**ステートメントは次のように書きます。

**DIM 配列名 (大きさ、大きさ、……)**

ここで大きさと書いたのは添字の最大値のことで、この値によって指定された大きさの記憶場所を確保します。

また配列宣言は省略することも可能です。

配列宣言を省略して配列変数を使った場合は

**DIM 配列名 (1 0)**

と宣言したのと同じになります。

## 3

## OPTION BASE

配列の添字の最小値は0か1です。

この値を決めるには **OPTION BASE**ステートメントを使用します。

**OPTION BASE { 0  
1 }**

**BASIC**起動時には、添字の最小値は0になっていますが**OPTION BASE 1**を宣言することによって添字の最小値を1に設定することができます。また、1度**OPTION BASE**ステートメントを宣言すると再度宣言することができません。

**OPTION BASE**を実行する場合は配列変数があらかじめすべて消去されている必要があります。

もし配列変数が存在した状態で **OPTION BASE**ステートメントを実行しようとするすると **Duplicated definition**のエラーとなります。



## 4

## 配列変数の消去

DIMステートメントで宣言した配列を消去するにはCLEARステートメントもしくは、ERASEステートメントを使用します。

CLEARステートメントを実行すると、変数はすべて消去されますが、このとき、配列変数もすべて消去されます。

ERASEステートメントは指定した配列変数の内容のみ消去し、次のように使います。

**ERASE** 配列名、配列名、……

## 5

## VDIM

DIMステートメントで宣言される配列変数は、メインメモリ上に記憶領域を確保します。

したがって、大量の配列を宣言すると、プログラムテキストとして使用する部分が少なくなり、Out of memoryエラーの原因となります。

そこで、大量のデータを扱う場合には、グラフィックVRAMの一部を変数領域として確保し、メインメモリにプログラムテキストを書く方法が便利です。

この場合、OPTION SCREEN 1 もしくは2で、グラフィックVRAMを、変数エリアとして使用するようあらかじめ定義しておきます。

前記のスクリーンモードで、グラフィックVRAMに配列変数（もしくは通常の変数）領域を確保する命令がVDIMステートメントです。

VDIMステートメントは

**VDIM** 配列名（大きさ、大きさ……）

のように書きます。

VDIMステートメントで宣言された配列変数は記憶領域がグラフィックVRAMに確保されること以外はDIMステートメントで宣言された配列変数と同様の使い方をします。

ただし、VDIMステートメントで宣言された配列変数はCLEARステートメントでは消去されません。

消去するには、VDIM CLEARというステートメントを用います。

また、ERASEステートメントはDIMの場合と同様に、VDIMで宣言された配列変数を消去することができます。

〔配列を使用したサンプルプログラム〕

```

10 WIDTH 40,25,0
20 INIT:CLS
30 OPTION SCREEN1
100 DIM A$(255,1)  ⇐配列データをメインメモリ上ではなくグラフィックRAM上にとる場合は、100行のDIM
110 FOR I=0 TO 255
115 CLS:LOCATE 0,0:PRINT HEX$(I)      100行のDIM A$(255,1)をVDIM A$(255,1)
120 FOR J=0 TO 1                        に変更します。
130 A$(I,J)=CGPAT$(J+1)*256+I)
150 PRINT
160 PRINT#0,A$(I,J):PAUSE 5
170 NEXT J
180 NEXT I

```

この例は2次元配列を利用したプログラムです。

100行で、A\$という配列を宣言しています。130行でキャラクタゼネレータのデータを読み出し、配列の中に入れていきます。配列 A\$(I, J)のうち J=0ときROMのデータが入り、J=1のときRAMのデータが入ります。160行で読み出した内容を表示しています。

以上の操作を256回くり返しています。

配列データをメインメモリ上ではなくグラフィックVRAM上にとる場合は、100行のDIM A\$(255, 1)をVDIM A\$(255, 1)に変更します。



# 11章

## 機械語サブルーチンとモニタ

ここでは、BASICプログラムを機械語サブルーチンとともに使用方法を説明します。

### 1

### 機械語サブルーチンの入力

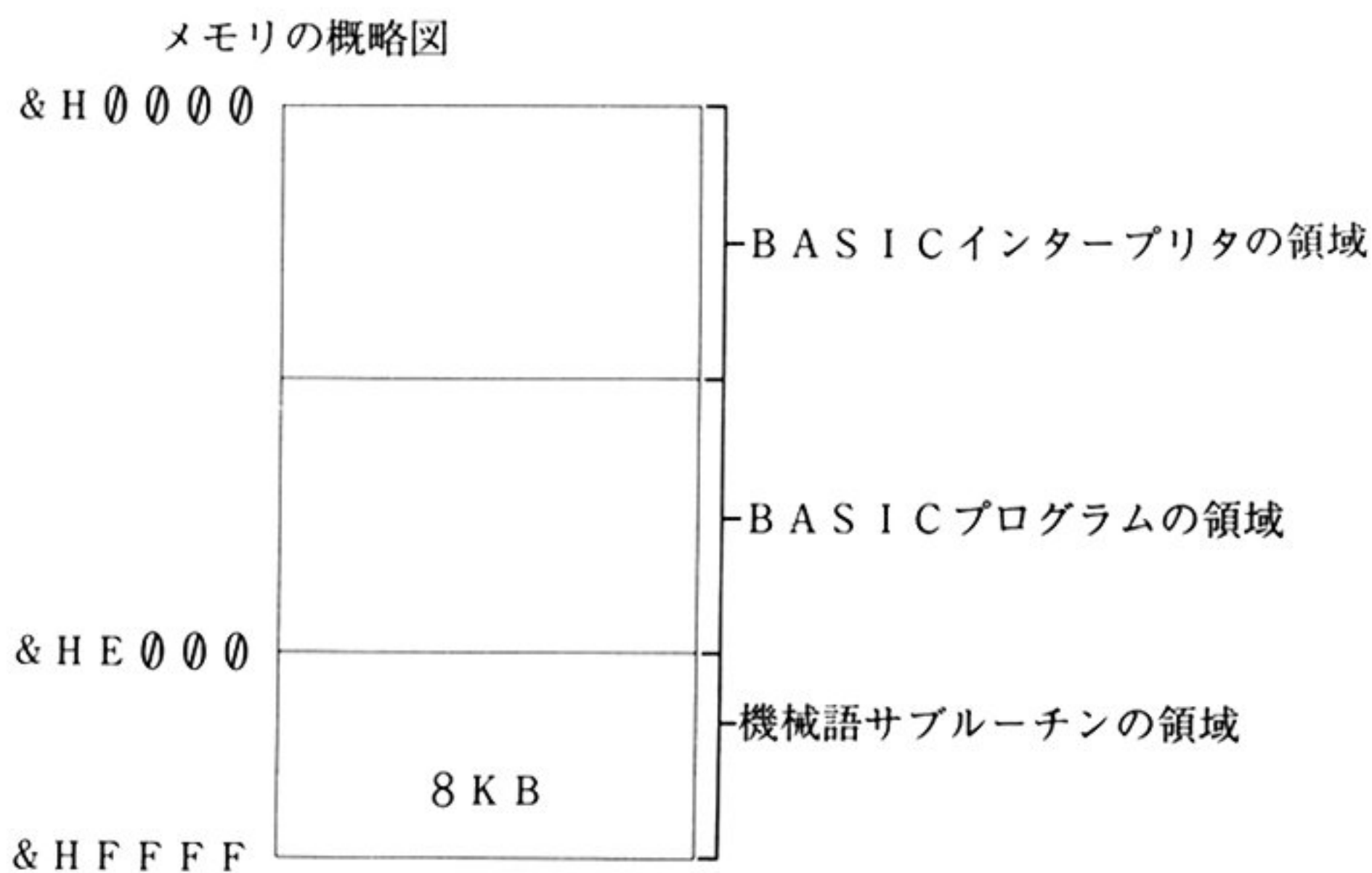
#### 1.1 機械語サブルーチンの記憶領域の設定

機械語サブルーチンを入力するためには、まずCLEARコマンドを用いて、その記憶領域を確保する必要があります。

たとえば、BASICの64KBのメインメモリ中、上部の8KB（最上部の3KBはワークエリアとして使用されるので実際は5KB）を確保したければ、

**CLEAR &HE000**

というステートメントをプログラムの先頭に書くか、そのままダイレクトに実行します。



# 11章

これで、BASICプログラムが機械語サブルーチンの領域に侵入することがなくなります。

ただし、最上部の&HF400～&HFFFFの3KBバイトは、BASIC、IPL（Initial Program Loader）およびBIOSのワークエリアとなっているため使用することができません。また、&HF000～&HF3FFは辞書のエリアとして使用されるので、辞書を使うときはこのエリアを使用することができません。

## 1.2 機械語サブルーチンの入力方法

機械語サブルーチンをメモリに記憶させる方法には、次の2通りがあります。

- (1) P O K Eステートメントを使う。
- (2) M O Nコマンドで機械語モニタを起動する。

### (1) P O K Eステートメントによる入力

P O K Eステートメントを使って、機械語サブルーチンを入力する手順を次に示します。

1. 機械語（16進数表現のコード）によってプログラムを作ります。
2. 機械語を、1バイトごとにカンマ（,）で区切って、D A T A文の中にデータとして用意します。
3. 機械語データをR E A Dステートメントで逐次読み込んで、P O K Eステートメントでメモリに書き込みます。

この方法は、比較的小さなプログラムを記憶するときに使います。

次に簡単な例を示します。

```
100 'P O K Eステートメントによる入力
110 CLEAR &HD000
120 I=0
130 READ D$
140 D=VAL("&H"+D$)
150 POKE &HD000+I,D
160 IF D=&HC9 THEN 190
170 I=I+1
180 GOTO 130
190 FOR J=&HD000 TO &HD000+I
200 RD=PEEK(J)
210 PRINT HEX$(RD)
220 NEXT
230 END
240 '機械語データ
250 DATA 3E,07 : 'LD A,07H
260 DATA 01,00,20 : 'LD BC,2000H
270 DATA ED,79 : 'OUT (C),A
280 DATA 3E,41 : 'LD A,41H
290 DATA 01,00,30 : 'LD BC,3000H
300 DATA ED,79 : 'OUT (C),A
310 DATA C9 : 'RET
```

このプログラムは、&H D 0 0 0番地から機械語データを入力しています。


1 1 0：機械語サブルーチンの記憶領域を&H D 0 0 0から確保しています。

1 3 0～1 8 0：2 5 0～3 1 0のD A T A文で指定された機械語データを読み  
P O K Eステートメントでメモリに書き込んでいます。

1 9 0～2 2 0：書き込んだデータをP E E K関数で読み出して表示しています。

### (2)機械語モニタによる入力

機械語モニタ（後述「③機械語モニタ」参照）を用いて、機械語サブルーチンを入力する方法を次に示します。

1. M O N と打ち込んで、機械語モニタを起動します。
2. モニタのDコマンドかMコマンドを実行し、メモリに機械語（16進数表現のコード）を書き込みます。
3. 必要に応じてSコマンドを使い、作った機械語サブルーチンをカセットテープに記録します。
4. Rコマンドで機械語モニタからB A S I Cに戻ります。



BASICから機械語サブルーチンを呼び出すには、CALLステートメントを使う方法と、USR関数を使う方法の2通りがあります。

## 2.1 CALLステートメント

機械語サブルーチンは、BASICのCALLステートメントを使って呼び出すことができます。CALLステートメントの形式は次の通りです。

```
CALL a
```

a：機械語サブルーチンの開始アドレス。

機械語サブルーチンの最後には、もとのプログラムに戻るためにC9（16進数、ニーモニックではRET）が置かれます。

CALLステートメントを実行すると、プログラム・カウンタの内容をスタックに退避し、CALLステートメント中に指示されたアドレスa、すなわちサブルーチンの入口へジャンプします。

機械語サブルーチンの処理を終えてC9に來ると、スタックに退避しておいたプログラムカウンタをもとに戻し、もとのBASICプログラムの実行を再開します。

CALLステートメントではBASIC上のデータを機械語サブルーチンへ引き渡す機能がありません。もし、データの引き渡しが必要な場合には次に述べるUSR関数を使用する方法が便利です。

## 2.2 USR関数

USR関数は、引数の値を受け渡すことができ、また、エラー処理のサブルーチンを呼び出すことができる、という利点を備えています。

USR関数を使うためには、まず、呼び出す機械語サブルーチンの実行開始アドレスを定義する必要があります。それにはDEFUSRステートメントを使って、次のように書きます。

```
DEFUSR n = a
```

n：USR関数識別番号。0～9の整数。

a：実行開始アドレス。

(例) 100 DEFUSR0=&HE000  
110 DEFUSR1=&HEC00

nは、関数識別番号で、0～9の最大10個の機械語サブルーチンを定義することができます。nを省略すると0が指定されます。

実行開始アドレスaは、機械語サブルーチンの実行が開始されるアドレスであり、メイン・メモリ64KB中のどこにでも指定できます。

DEFUSRでいったんアドレスを定義すると、再度定義しなおすまで、そのアドレスが保持されます。

DEFUSRで、nとaを定義したら次のようにしてUSR関数を使うことができます。

- [1]  $y = \text{USR } n(x)$   
[2]  $y\$ = \text{USR } n(x\$)$

- [1] y：数値変数。  
n：USR関数識別番号。0～9の整数。  
x：数式（引数）。  
[2] y\$：文字変数。  
n：USR関数識別番号。0～9の整数。  
x\$：文字式（引数）。

(例) 200 A=USR0(65)  
210 A\$=USR1(C\$)

USRの後ろにつける番号nは、DEFUSRで設定した番号に対応しています。

USR関数は、引数xやx\$の値をもって、n番の機械語サブルーチンを呼び出し、リターン時にその値をyやy\$に代入します。機械語サブルーチンの中で、引数の値を書き換えると、その結果がyやy\$の値となります。

USR関数の引数x、x\$は省略することができません。

xは、数式で、単独の定数、数値変数でもかまわず、精度も整数型、単精度型、倍精度型のいずれでもかまいません。

x\$は、文字列を値にもつ文字式です。

USR関数で機械語サブルーチンを呼び出すとき、CPUの各レジスタには次のような値が入っています。

#### ●Aレジスタ（アキュムレータ）

Aレジスタ（アキュムレータ）には、引数の型を示す値が入っています。

引数の型	Aレジスタの値
整数型	2
単精度型	5
倍精度型	8
文字型	3

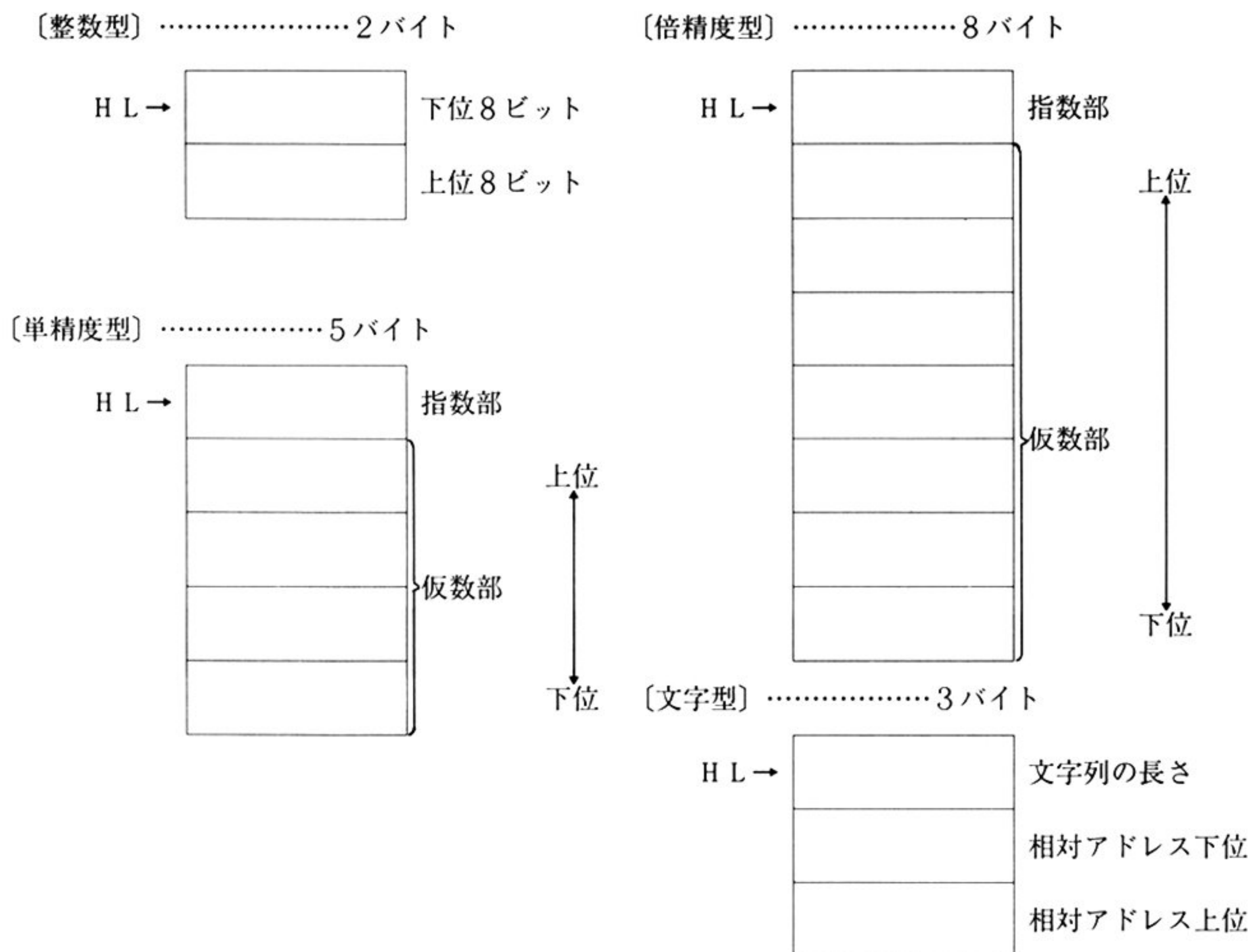
これらの値は、文字変数を除き、メモリ内において何バイトで表現されているかを表わしています。

#### ●HLレジスタ

HLレジスタの値は、数値変数と文字変数の場合で異なります。数値変数の場合は引数の入っているメイン・メモリのアドレスを示しており、引数が文字変数の場合は、直接文字列の入っているアドレスを示さず、ストリング・ディスクリプタと呼ばれている相対アドレスを示しています。ストリング・ディスクリプタは、メモリの文字変数領域の先頭アドレスから何番地目に文字列が入っているのかを示す値で、直接文字列の入っているアドレスを示すものではありません。文字列の入っているアドレスは、DEレジスタに入っています。

引数の値は、HLレジスタが指しているアドレスから次のような形式で格納されています。





### ● D E および B レジスタ

D E と B レジスタは、A レジスタが 3 のとき、すなわち U S R 関数の引数が文字変数のときのみ意味をもちます。

このとき、D E レジスタにはその文字データが格納されているアドレス（絶対アドレス）、B レジスタには文字データの長さが入っています。

U S R 関数の引数が文字変数だけのときは、引数の格納されているアドレスと U S R 関数実行後の引数のアドレスが同じなので、引数の値が変わってしまう可能性があります。したがって、文字変数を引数にするときは、

**U S R (A \$ + " ")**      ※ " " はヌルストリングです。

というように、ヌルストリングを連結した形にしてください。これで、引数 A \$ の値が U S R 関数実行前と実行後とで変わることはありません。

### ● U S R 関数内でのエラー処理

U S R 関数の中でエラーが発生した場合、B A S I C にエラーの発生を知らせることができます。このとき、B A S I C 内であらかじめ O N   E R R O R   G O T O の処理をしていれば、エラー処理ルーチンへジャンプすることもできます。

その方法は、エラー番号を A レジスタに入れ I X レジスタの示すアドレスにジャンプすることにより行なうことができます。


(例) 機械語サブルーチン中にエラーが発生したら、Type mismatch (エラー番号13) の処理へジャンプする場合、

```
      :  
      :  
      :  
      J P  Z, TYPEMS .....エラーが発生したら分岐します。  
      :  
      :  
      :  
      R E T  
TYPEMS: L D  A, 13 .....エラー番号13をAレジスタへいれます。IXレジ  
      J P  (IX)          スタの示すアドレスへジャンプします。
```


### 3

## 機械語モニタ

本機は機械語の扱いを容易にするため、簡単な機械語モニタを備えています。  
機械語モニタを起動するには、BASICのMONコマンドをダイレクトに入力してください。

```
      :  
      O k  
      M O N   
      *
```

MONコマンドを入力すると、上のように\*印が表示され、機械語モニタが起動します。

また、本機の電源をONにするかBASICでBOOTを実行して、IPLを起動しているとき **SHIFT** + **BREAK** を押しつづけると、次の画面となります。

この画面のとき、**M**キーを押しても機械語モニタを起動することができます。ただし、この場合はモニタのコマンドのRで機械語モニタから抜け出すことができません。

Make your device ready

Press selected key to start driving:

F : F l o p p y

R : R O M

C : C M T

T : T i m e r



## 機械語モニタのコマンド

機械語モニタには、次に示す14種類のコマンド用意されています。

コマンド	働 き
D	D u m p m e m o r y (メモリダンプ)
M	s e t M e m o r y (メモリセット)
F	F i n d d a t a (データサーチ)
P	P r i n t e r s w i t c h (プリンタスイッチ)
G	G o s u b
T	T r a n s f e r d a t a (データ転送)
S	S a v e
L	L o a d
V	V e r i f y
R	R e t u r n
!	B I O S R O M / M A I N R A M a c c e s s m o d e s w i t c h
#	W I D T H 4 0 / 8 0 s w i t c h、画面の初期化など
W	W r i t e (各デバイスへ)
Y	r e a d (各デバイスから)

### D (dump memory)

働き：メインメモリの内容を表示します。

文法：\*D [XXXX [YYYY]]

XXXX：先頭アドレス。16進数4けた以内。0～FFFF。

YYYY：最終アドレス。16進数4けた以内。0～FFFF。

説明：XXXXからYYYYまでのメインメモリの内容を表示します。(例として、40文字モード時の表示について示します)

YYYYを省略すると、XXXXから128バイト分(8バイト×16行)表示します。XXXXもいっしょに省略すると、それまで表示した次のアドレスから128バイト分表示します。


画面の1行分の表示形式は、次のようになります。

：XXXX=HH HH HH HH HH HH HH HH/CCCC. C.

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑  
(1) (2) (3) (4) (5) (6)


(1) 編集可能なことを示すマーク。

(2) 先頭データのアドレス。16進数4けた。0000～FFFF

- この状態のとき、次の〔編集書式〕に従って、メインメモリの内容を書き換えることができます。表示のストップは **BREAK** キー、再開はスペースキーか  キーを押します。また、コマンド待ちに戻るには **SHIFT** + **BREAK** キーを押します。

機械語モニタもBASICと同様にスクリーン・エディタを備えており、メインメモリのダンプ後、あるいはサーチ後、次の書式に従って内容を書き換えることができます。

$$\begin{array}{ccccc} \uparrow & & \uparrow & \uparrow & \uparrow & & \uparrow & \uparrow \\ (1) & (2) & (3) & (4) & (5) & & (4) & (5) \end{array}$$

- 書き換えた後、その行で  キーを押すと、その行のデータがメモリに記憶されます。






## M (set memory)

働き：メインメモリの内容を1バイト表示します。


文法：\*M [ XXXX ]

XXXX：先頭アドレス。16進数4けた以内。0～FFFF。

説明：MXXXX  と入力すると、そのアドレスの内容を表示して、データの入力待ちとなります。

```
*MFE00   
:FE00=H
```

↑  
カーソル

このとき、前述の〔編集書式〕に従って、データを書き換えることができます。書き換えた後、キーを押すと、データがメモリに記憶され、記憶されたデータ数から次のアドレスが計算されて、改行されます。

```
*MFE00   
:FE00=3E 41 13 00 C9   
:FE06=H
```

↑  
カーソル

この状態から抜け出すには、**SHIFT** + **BREAK** キーを押します。

## F (find data)

働き：データを探し出します。


文法：\*F XXXX YYYY HH HH .....

XXXX：先頭アドレス。16進数4けた以内。0～FFFF。

YYYY：最終アドレス。16進数4けた以内。0～FFFF。

HH：データ。16進数2けた以内。0～FF。

説明：XXXXからYYYYまでのメインメモリから、HH HH .....で指定された一続きのデータ群を探し出し、見つかったときは、そのアドレスとデータを表示します。

```
*F1000 2000 CD 41 00   
:10C3=CD 41 00 /へA.
```

\*■

↑  
カーソル

上の例は、「CD 41 00」という3バイトのデータを、メモリの1000～2000番地（16進数）から探して、10C3番地にそれが見つかったことを示しています。

このコマンドを実行すると、該当するアドレスとそのデータを、あるだけ何行でも表示します。

途中で、このコマンドから抜け出たいときは、**SHIFT** + **BREAK** キーを押してください。

### **P (printer switch)**

**働き：** 画面表示とプリンタ表示の切り換えをします。

**文法：** \*P

**説明：** Pコマンドを実行すると、DおよびFコマンド実行後の表示を、プリンタにしたり、画面にしたり、切り換えることができます。

機械語モニタを起動した時点では、画面に表示するようになっていますが、このコマンドを入力するたびに、プリンタ表示／画面表示、と表示するデバイスが反転します。

プリンタが正しく接続されていない場合は、しばらくしてから「Err 73」と画面に表示されてコマンド待ちになります。そのときは、プリンタをチェックするか、再度Pコマンドを実行して表示を画面に戻してください。

### **G (gosub)**

**働き：** 機械語サブルーチンを呼び出します。

**文法：** \*G HHHH

HHHH： アドレス。16進数4けた以内。0～FFFF

**説明：** メインメモリ上のHHHから始まるサブルーチンを呼び出します。

### **T (transfer data)**

**働き：** データの転送をします。

**文法：** \*T XXXX YYYY ZZZZ

XXXX： データの先頭アドレス。16進数4けた以内。0～FFFF

YYYY： データの最終アドレス。16進数4けた以内。0～FFFF

ZZZZ： データの転送先のアドレス。16進数4けた以内。0～FFFF

**説明：** メインメモリ上のXXXXからYYYYまでのデータをZZZZを先頭とする場所に転送します。

### **S (save)**

**働き：** メインメモリの内容をカセット・テープに記録します。

**文法：** \*S XXXX YYYY ZZZZ : file name

XXXX： 先頭アドレス。16進数4けた以内。0～FFFF

YYYY： 最終アドレス。16進数4けた以内。0～FFFF

ZZZZ： 実行開始アドレス。16進数4けた以内。0～FFFF

**説明：** XXXXからYYYYまでのメインメモリの内容をカセットテープに記録します。

再びLコマンドでロードするとき、先頭アドレスが省略されると、XXXXからロードされます。

ZZZZは、IPLから機械語プログラムを走らせたり、BASICからLOADMコマンドのRオプションによって走らせるときに、実行の開始アドレスとなります。

```
*S  0000  1FFF  0000 : TEST.BIN
      ↑      ↑      ↑      ↑
      (1)    (2)    (3)    (4)
```



- (1) ロード時の先頭アドレス
- (2) ロード時の最終アドレス
- (3) 実行開始アドレス
- (4) ファイル名

file name は、13文字以内のファイル名と3文字以内のエクステンションからなり、  
:の後ろに続けて書きます。

## L (load)

働き： カセットのデータをメインメモリに読み込みます。

文法： \*L [XXXX] [:file name]

XXXX： ロード先頭アドレス。16進数4けた以内。0~FFFF

説明： カセット・テープに記録されているデータや機械語プログラムをメインメモリに読み込みます。

XXXXを指定すると、そのアドレスから読み込み、省略すると、Sコマンドで記録した通りの形で読み込みます。

file nameを省略すると、最初に見つけたファイルを読み込みます。

読み込んでいる途中で、**SHIFT** + **BREAK** キーを押したり、その他のエラーが生じたときは、「Err 29」と表示されコマンド待ちの状態に戻ります。

このコマンドは機械語プログラムを読み込むだけで実行は行ないません。

このコマンドによって、読み込んだ機械語プログラムを実行するには、Gコマンドを使います。

## V (verify)

働き： カセットの内容とメインメモリの内容を比較します。

文法： \*V [:file name]

説明： 指定したファイルをカセット・テープから読み込み、メインメモリの内容と比較します。

Sコマンドで機械語プログラムやデータが正しく記録されたかどうか調べるときに使います。もしカセット・テープから読み込んだデータとメモリの内容と一致していなければ、「Err 29」を出して止まります。

「Err 29」が出たときは、再度Sコマンドで記録しなおしてください。

## R (return)

働き： 機械語モニタからシステムに戻ります。

文法： \*R

説明： 機械語モニタを起動したBASICに戻ります。

このとき、SP (スタック・ポインタ) およびHLレジスタの内容は保存されているので、MONの次のコマンドやステートメントに移ります。次の命令がないときは、コマンド待ちの状態となります。

\*R

Ok



↑

カーソル


! (change access mode)

働き: IPL ROMとメインメモリのアクセスの切り換えをします。

文法: \*!  
)!

説明: モニタのコマンド (D、M、F、G、T) でアクセスされるメモリ、バンクをIPL ROMにするか、メインメモリにするかを切り換えることができます。

機械語モニタを起動した時点では、メインメモリがアクセスされており、コマンド待ちのとき「\*」が表示されますが、

\*! 

を実行すると、IPL ROMがアクセスされるようになり、コマンド待ちのとき「)」が表示されます。

また、その状態で

)!

を実行すると、「\*」が表示されメインメモリのアクセスに戻ります。

IPL ROMのアクセス時、Dコマンドによる読み込むメモリの表示はIPL ROMになっていますが、書き込むメモリはあくまでもメインメモリになるので注意してください。

# (set screen mode)


働き: 画面モードの切り換え、PALETの初期化、グラフィック画面の消去をします。

文法: [1] \*#  
[2] \*#P  
[3] \*#C  
[4] \*#kdm


k: 0または1。

d: 0~2。

m: 0~5の整数。

説明: [1] \*# 



WIDTH 40とWIDTH 80の切り換えをします。


[2] \*#P 

PALETの初期化をします。

PALET 0, 0: PALET 1, 1: PALET 2, 2

: PALET 3, 3: PALET 4, 4: PALET 5, 5

: PALET 6, 6: PALET 7, 7  または PALET @ 0, 1, 2,  
3, 4, 5, 6, 7  と同じ

[3] \*#C 

グラフィック画面の消去をします。

(=CLS 0 )



[4] \*# k d m

k、d、m の値によって次のような設定ができます。

k の値	コードの指定
0	16進数表現のキャラクタコード80～9F、E0～FFを半角文字（1バイトコード文字）のコードとします。 (=KMODE 0)
1	キャラクタコード80～9F、E0～FFを全角文字（2バイトコード文字）のコード（シフトJISコード）の第1バイトとします。 (=KMODE 1)

d の値	使用モニター
0	専用ディスプレイテレビの標準／高解像度切換えスイッチの状態に従います。
1	標準ディスプレイの選択
2	高解像度ディスプレイの選択

m の値	設定される画面
0	テキスト画面25行／グラフィック画面200ライン
1	テキスト画面12行／グラフィック画面192ライン
2	テキスト画面20行／グラフィック画面なし
3	テキスト画面10行／グラフィック画面なし
4	テキスト画面25行／グラフィック画面400ライン
5	テキスト画面12行／グラフィック画面384ライン

[例] KMODE 1、標準ディスプレイ、テキスト画面10行に設定します。

\*# 1 1 3 (パラメータ間にスペースを入れてはいけません。)

## W (device Write)

働き： メインメモリの内容を指定デバイスの指定レコードに書き込みます。

文法： \*WX d:nnnn rr aaaa

X： デバイス名。

M……………グラフィック・メモリ (d = 0 または 1)

E……………外部メモリ (d = 0～9)

D……………3または5インチ版ディスク (d = 0～3)

F……………8インチ版ディスク (d = 0～3)

H……………5インチ10Mハードディスク (d = 0～3)

d: ドライブ番号。(デバイス名Xによって範囲が異なります)  
 nnnn: 書き込む先頭のレコード番号。最大4けたの16進数。  
 rr: 書き込むレコード長<sup>1)</sup>。1~FFの16進数。  
 aaaa: メモリの先頭アドレス。0~FFFFの16進数。

1) レコード長 1単位が1セクタ(標準256バイト)。

説明: Xで指定されたデバイスのd番ドライブのレコード番号nnnnに、メインメモリ内のアドレス&Haaaaから始まるレコード長rrのデータを書き込みます。

デバイスXにDかFでディスクを指定するときは、Mコマンドを実行して、ディスクのタイプを示すデータをメモリのワーク・エリアに書き込んでおく必要があります。

次に、使用するディスクのドライブ番号と書き込むアドレスの関係とフロッピー・ディスクと書き込むデータの関係を示します。

使用するドライブ番号		ディスクのタイプを表わすデータを書き込むアドレス
3または5インチ版	ドライブ0	FAB4
	1	FAB5
	2	FAB6
	3	FAB7
8インチ版	ドライブ0	FAB8のビット1、0
	1	FAB8のビット3、2
	2	FAB8のビット5、4
	3	FAB8のビット7、6


フロッピーディスクのタイプ	書き込むデータ
3または5インチ版 2D (320K)	0
5インチ版 2DD (640K)	1
5インチ版 2HD (1M)	2
5インチ版 *2HD (1M、標準フォーマット)	3
8インチ版 2D-256 (1M)	00
8インチ版 *2D-256 (1M、標準フォーマット)	01
8インチ版 *1S-128 (240K)	10

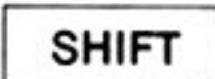

(8インチディスクのデータは2ビット毎のビットパターン1バイトから成っています。)




(例) 3または5インチのドライブ0で2DDタイプのフロッピーディスクを使用する場合

\*M F A B 4 

: F A B 4 = 0 1 

: F A B 5 = 0 0  + 

\*W D 0 : 1 0 8 C 0 0 0  ..... 3または5インチ版ディスクドライブ0のフロッピーディスク上のレコード番号1以降に、メインメモリ内のアドレス&H C 0 0 0 から8レコード分のデータを書き込みます。

## Y (device read)

働き: 指定デバイスの指定レコードから指定のメインメモリアドレスにデータを読み込みます。

文法: \*Y X d : n n n n r r a a a a

X: デバイス名。

M ..... グラフィック・メモリ (d = 0 または 1)

E ..... 外部メモリ (d = 0 ~ 9)

D ..... 3 または 5 インチ版ディスク (d = 0 ~ 3)

F ..... 8 インチ版ディスク (d = 0 ~ 3)

H ..... 5 インチ 10M ハードディスク (d = 0 ~ 3)

d: ドライブ番号。(デバイス名Xによって範囲が異なる)

n n n n: 読み込む先頭のレコード番号。最大4けたの16進数。

r r: 読み込む<sup>1)</sup>レコード長。1 ~ F F の16進数。

a a a a: メモリの先頭アドレス。0 ~ F F F F の16進数。

1) レコード長 1単位が1セクタ(標準256バイト)。

説明: Xで指定されたデバイスのd番ドライブのレコード番号n n n nから、メインメモリ内のアドレス&H a a a aから始まるエリアに、レコード長r rのデータを読み込みます。

デバイスXにDかFでディスクを指定するときは、Mコマンドを実行して、ディスクのタイプを示すデータをメモリのワーク・エリアに書き込んでおく必要があります。書き込むデータについてはWコマンドを参照してください。

# 12章

## ディスクの使い方

本機では、5インチミニフロッピーディスク（3インチコンパクトフロッピーディスク）のほか、に8インチフロッピーディスク、5インチハードディスクをBASICでサポートしています。ここではこれらのフロッピーディスク装置を入力出力装置として使う場合に、必要な事柄を説明しています。

なお、プログラムやデータを保存または再生するときは「8章 プログラム、データファイルの保存と再生」をご覧ください。

### 1

### ディスクの使用準備

新しいフロッピーディスクを使用される場合、あらかじめフロッピーディスクのフォーマットングを行なっておく必要があります。

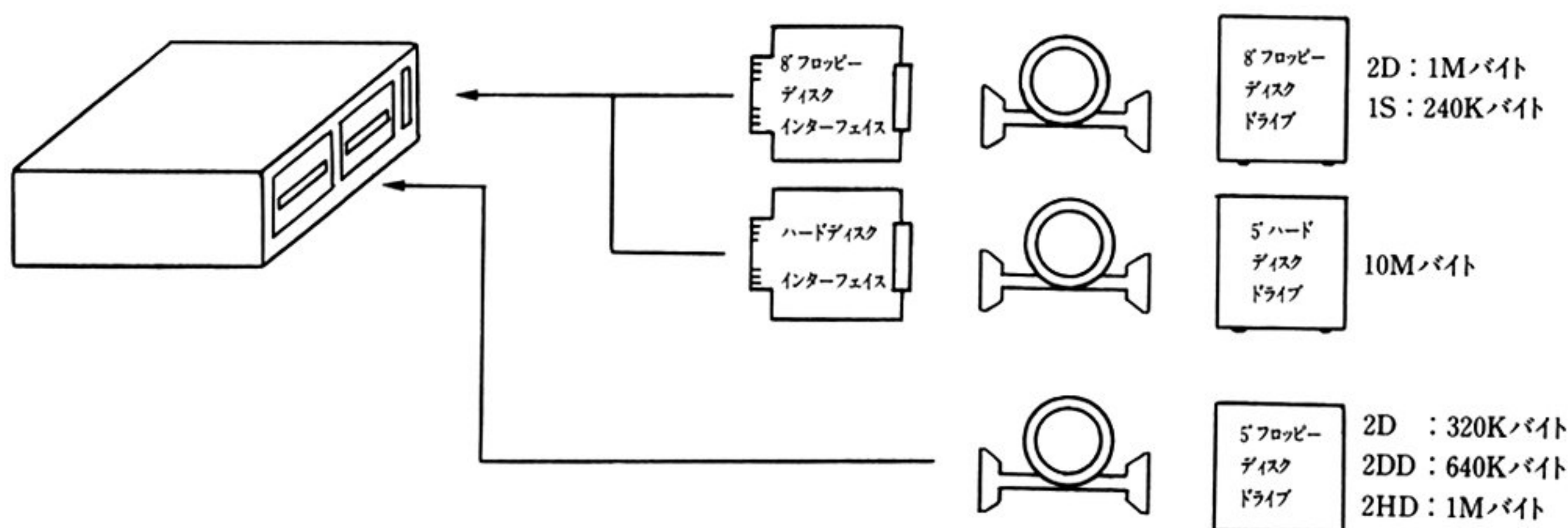
フォーマットングとは新しいフロッピーディスクを使用できるようにすることで、DISK BASICのユーティリティプログラムののためのプログラムが用意されていますので、フロッピーディスクのフォーマットングの手順に従って行なってください。

（別冊の『アプリケーションソフトの説明書』の「ディスクユーティリティ」参照）

### 2

### ドライブ接続構成

本機は下記構図の様に各々3インチまたは5インチフロッピーディスクドライブ、（容量：320Kバイト、640Kバイト、1Mバイト）、8インチフロッピーディスクドライブ（容量：1Mバイト、240Kバイト）および5インチハードディスクドライブ（容量10Mバイト）を接続可能です。



ドライブ接続構成図



本機は5インチフロッピーディスクドライブ及びインターフェイスを内蔵しているのので3インチまたは5インチフロッピーディスクドライブは直接増設可能であり、8インチフロッピーディスクドライブ、ハードディスクドライブは専用インターフェイスボードを使用して外部接続ができます。

〈ドライブ接続台数〉

接続ドライブ		台 数	備 考
3インチ または 5インチ フロッピー ディスク ドライブ	内 蔵	2	最大接続台数はトータルで 4台
	内部増設	—	
	外部増設	2	
8インチフロッピー ディスクドライブ		4	専用インターフェイス使用
5インチハードディスク ドライブ		4	専用インターフェイス使用

(サポートドライブを最大接続した時の容量は48Mバイトです。)

本機には外部5インチフロッピーディスクインターフェイスは接続しないでください。

### 3

## システムプログラム起動方法

本機では、3インチまたは5インチフロッピーディスクドライブをトータル4ドライブ、8インチフロッピーディスクドライブを4ドライブ（専用インターフェイス使用）およびハードディスクドライブを4ドライブ（専用インターフェイス使用）が一度に接続可能であり、8種類のディスクタイプ（次表参照）をサポートしているのでシステムプログラムを起動する場合あらかじめシステムディスクの入っているドライブナンバーとディスクタイプを設定する必要があります。その方法として

- (1)初期モードスイッチ（ディップスイッチ）により、システムディスクのタイプ設定
- (2)キー入力により、システムディスクのタイプ設定

の2種類が可能です。

#### (1)初期モードスイッチによるシステムディスクのタイプ設定

この方法は、初期モードスイッチにより起動するシステムディスクナンバーを設定し、システム電源を投入することによってサポートドライブ（3インチまたは5インチおよび8インチフロッピーディスクドライブ、ハードディスクドライブ）のドライブ0から自動的にシステムプログラムが起動できます。

＊出荷時ディップスイッチは全てON（ディスクタイプナンバー0）に設定しています。

ディスクタイプナンバーとディスクの種類

ディップスイッチ			ディスク タイプ ナンバー	内 容	記録方式	記録内容	ディスク名
SW1	SW2	SW3					
ON	ON	ON	0	2D X1 フォーマット	両 面 倍密度記録	3 2 0 K	3インチ 5インチFD
OFF	ON	ON	1	2DD X1 フォーマット	両面倍トラック 倍密度記録	6 4 0 K	
ON	OFF	ON	2	2HD X1 フォーマット	両面高密度 倍密度記録	1 M	
OFF	OFF	ON	3	*2HD 8インチ 標準フォーマット	両面高密度 倍密度記録	1 M	
ON	ON	OFF	4	2D X1 フォーマット	両 面 倍密度記録	1 M	8インチFD
OFF	ON	OFF	5	*2D 8インチ 標準フォーマット	両 面 倍密度記録	1 M	
ON	OFF	OFF	6	1S 8インチ 標準フォーマット	片 面 単密度記録	2 4 0 K	
OFF	OFF	OFF	7	X1 フォーマット	4ヘッド 倍密度記録	1 0 M	
							ハードディスク

\*但しヘッド0、シリング0のみ1セクタ=128バイトの単密度記録

またシステムプログラムを起動したサポートドライブ0に対するディスクタイプも同時に、システムのディスクタイプ設定レジスタに書き込まれ、以後のプログラム上で、ドライブのデバイス名を省略してアクセスしようとした場合は、このドライブナンバーが指定されます。

一般にこの様にシステムプログラムを起動したドライブを、**カレントドライブ**と呼んでいます。

一方、システムプログラムが起動できなかった時は

```

Make your device ready
Press selected key to start driving:
F : Floppy
R : ROM
C : CMT
T : Timer

```

という表示となるので初期モードスイッチで設定したディスクタイプのドライブ状態の再確認後、再び電源を入れてシステムプログラムを起動するか、キーボード入力での操作で、システムプログラムを起動してください。



## (2)キー入力でシステムプログラムを起動する場合

初期モードスイッチの設定に関係なくキー入力によって、ドライブの選択を行ない任意のディスクタイプ（ナンバー0から7）のディスクから、システムプログラムが起動できます。またそのドライブナンバーがカレントドライブに設定され、デバイス名を省略してアクセスした時はカレントドライブ（システムプログラムを起動したドライブ）が指定されます。

### 操作方法

本機の電源をONすると

```
Make your device ready
Press selected key to start driving:
      F : Floppy
      R : ROM
      C : CMT
      T : Timer
```

.....(1)

という表示が画面に出ますので、キーボードから[F]キーを入力します。

```
Drive No ?      (0—3)
```

.....(2)

という表示となりますので、次にシステムプログラムを起動しようとするドライブナンバー（0～3）を指定するために、数字キーの[0]～[3]のうち指定のナンバーをキー入力すると、

```
Type of DISK ?      (0—7)

5 " FD  320 kbytes    2 D      ..... 0
or
3 " FD  640 kbytes    2 DD     ..... 1
          1 M   bytes    2 HD     ..... 2
          1 M   bytes    *2 HD    ..... 3
                                           .....(3)

8 " FD  1 M   bytes    2 D—256   ..... 4
          1 M   bytes    *2 D—256 ..... 5
          240 kbytes    *1 S—128   ..... 6

5 " HD  10 M bytes          ..... 7
```

とサポートしている全てのディスクタイプが表示されますので、起動したいディスクタイプに対応しているナンバーを指定するために、数字キーの[0]～[7]キーを使って入力すると任意のドライブナンバーから任意のディスクタイプでシステムプログラムが起動されます。表示画面は

```
IPL is lookig for a program from FD [NO] .....(4)
                                           (NOは0～3のうちどれか1つ)
```

となり、設定したディスクタイプで読み出し可能な時は

表示となり、BASICの読み出しを開始します。

一方設定したドライブナンバーから、設定したディスクタイプで読み出し不可能だった時は(1)の表示になるので、入出力装置の状態を確認の上、再操作してください。また、(2)または(3)の画面の時 **SHIFT** + **BREAK** キーを押しますと、(1)の表示画面に戻すことができます。入力ミスをした場合などに利用してください。

キーボードから設定して、システムプログラムを起動した場合、起動したドライブに対するディスクタイプのナンバーは自動的にシステムのディスクタイプ設定レジスタに書き込まれます。他のサポートドライブに対しては初期設定(3インチまたは5インチドライブの場合ディスクタイプ0、8インチドライブの場合はディスクタイプ4) されているので、各ドライブに対してディスクタイプを設定する必要があります。

## 4

## 接続ドライブのディスクタイプ設定

本機では、計12ドライブと接続が可能です。

	デバイス名
・ 3インチまたは5インチ(2D、2DD、2HD).....	4ドライブ ; 0 : ~ 3 :
・ 8インチ(2D、1S) .....	4ドライブ ; F0 : ~ F3 :
・ ハードディスク .....	4ドライブ ; HD0 : ~ HD3 :

3インチまたは5インチおよび8インチのフロッピーディスクドライブにどのタイプのディスクが挿入されているか、本機で認識する必要があります。本機で認識しているディスクタイプと違うディスクを挿入すると、書き込み／読み出しができない場合が生じます。すなわちディスクタイプによってヘッド数、シリング数、セクタ数が異なり、また、FM記録／MFM記録、3インチまたは5インチ(2D、2DD)／5インチ(2HD)、8インチ(2D、1S)の切換えを各フロッピーインターフェイスに指示する必要があるためです。

ディスクタイプを正確に認識することによって、自動的にプログラムの書き込み／読み出しの管理ができます。


ディスクタイプの設定方法として

- (1) ユーティリティプログラムの実行
- (2) DEVICE命令
- (3) ワークレジスタの書き換え

の3方法があります。

(1)は、ユーティリティプログラムを走らせ各デバイス名に対応するディスク・タイプNo. をキーボードから入力すると自動的に設定できます。

(2)は、BASICの DEVICE命令で

DEVICE "デバイス名：ディスクタイプNo." 

によって設定できます。(次表参照)

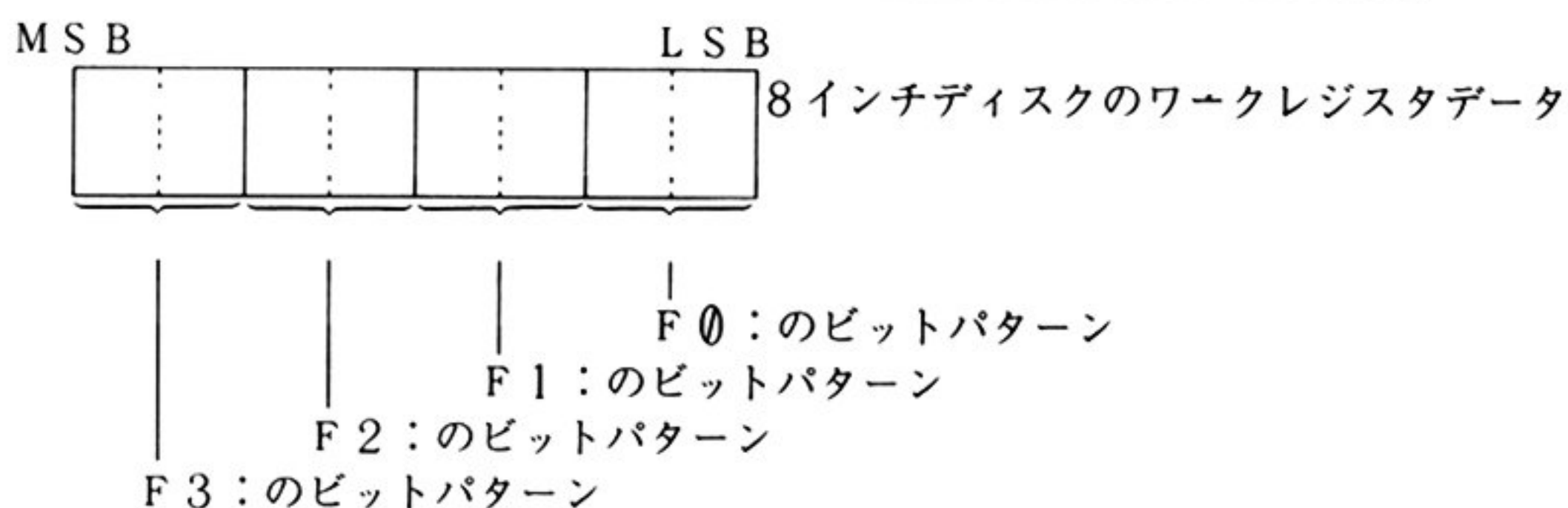


	デバイス名	ディスクタイプNo
3 インチ または 5 インチ F D	0 :	0... 2 D
	1 :	1... 2 D D
	2 :	2... 2 H D
	3 :	3... 2 H D
8 インチ F D	F 0 :	0... 2 D
	F 1 :	1... 2 D
	F 3 :	2... 1 S

(3)はモニタコマンドまたはBASICで、ワークレジスタを変更する方法です。

ディスクタイプのワークレジスタは、各デバイス名について5インチディスクは1バイト、8インチディスクは2ビット毎のビットパターン1バイトから成っています。

	デバイス名	ワークレジスタアドレス	データ
5 インチ F D	0 :	F A B 4	0 : 2D X1フォーマット
	1 :	F A B 5	1 : 2DD X1フォーマット
	2 :	F A B 6	2 : 2HD X1フォーマット
	3 :	F A B 7	3 : 2HD 8インチ標準フォーマット
8 インチ F D	F 0 :	F A B 8	0 0 : 2D X1フォーマット
	F 1 :		0 1 : 2D 8インチ標準フォーマット
	F 2 :		1 0 : 1S 8インチ標準フォーマット
	F 3 :		(ビットパターン)



例えば 接続ディスクが

3 インチまたは5 インチ F D	0 : 2 D	8 インチ F D	F 0 : 2 D X 1 フォーマット
	1 : 2 D		F 1 : 2 D X 1 フォーマット
	2 : 2 D D		F 2 : 2 D 8 インチ標準フォーマット
	3 : 2 H D X 1 フォーマット		F 3 : 1 S 8 インチ標準フォーマット

の場合

ディスクタイプのワークレジスタは

F A B 4	;	0 0
F A B 5	;	0 0
F A B 6	;	0 1
F A B 7	;	0 2
F A B 8	;	9 0

となります。

ディスクタイプのデフォルト（初期モードスイッチの設定の場合）

(1) 3 インチまたは5 インチディスクでB O O Tした時

デバイス名	デフォルトタイプ
0 :	B O O Tしたディスクタイプ
1 : ~ 3 :	2 D
F 0 : ~ F 3 :	2 D X 1 フォーマット

(2) 8 インチディスクでB O O Tした時

デバイス名	デフォルトタイプ
0 : ~ 3 :	2 D
F 0 :	B O O Tしたディスクタイプ
F 1 : ~ F 3 :	2 D X 1 フォーマット

(3) ハードディスクでB O O Tした時

デバイス名	デフォルトタイプ
0 : ~ 3 :	2 D
F 0 : ~ F 3 :	2 D X 1 フォーマット

また、キー入力で、B O O Tした時は、B O O Tしたドライブのデバイス名が、そのディスクタイプに設定され、他の3 インチまたは5 インチ、8 インチフロッピーディスクドライブは

3 インチまたは5 インチF D… 2 D

8 インチF D… 2 D X 1 フォーマット

に設定されます。

## 5

## ディスクのフォーマット構成

ディスクにデータの書き込み読み出し動作を行なう場合、ディスク上のアドレス情報（ヘッドN o.、シリンダーN o.、セクターN o等）を基にして、データの格納場所を検索し実行します。こ



のため、ディスク上にこのアドレスを割り付ける必要があります。この動作をフォーマット（1次フォーマット）といいます。

3インチまたは5インチFD 2D、2DDの市販ディスクはフォーマットされていないディスクなので、必ずフォーマットする必要があります。一方、5インチFD 2HDおよび8インチFD 2D、1Sは8インチFD標準フォーマットでフォーマットされているので、必ずしもフォーマットする必要はありませんが全レコード256バイト/セクター、倍密度記録のX1フォーマット注1)に、フォーマットして使用してもかまいません。

サポートしているディスクタイプおよびフォーマットを下記に示します。

項目 ディスク		ヘッドNo	シリンダNo	セクターNo	セクター データ (バイト)	内 容
3インチ または 5インチ FD	2D	0,1	0~39	1~16	256	X1 フォーマット
	2DD	0,1	0~79	1~16	256	X1 フォーマット
	2HD	0,1	0~76	1~26	256	X1 フォーマット
		0,1	0~76	1~26	256 注2) ヘッド0 シリンダー0 128	8インチFD 標準フォーマット
8インチ FD	2D	0,1	0~76	1~26	256	X1 フォーマット
		0,1	0~76	1~26	256 注2) ヘッド0 シリンダー0 128	8インチFD 標準フォーマット
	1S	0	0~76	1~26	注3) 128	8インチFD 標準フォーマット
ハードディスク		0~3	0~305	1~33	256	X1 フォーマット

注1) X1フォーマットは、全てのシリンダーに対して  
256バイト/セクターの倍密度記録

注2) 8インチFD標準フォーマットは  
ヘッドNo. 0、シリンダーNo. 0のセクターNo. 1~26は、  
128バイト/セクターの単密度記録  
その他の領域は  
256バイト/セクターの倍密度記録

注3) 全てのシリンダに対して  
128バイト/セクターの単密度記録

また、システムでディスク管理を行なうためにはディスク上にある特別な領域、FAT領域とディレクトリ領域を設け、システムに合うようにFAT領域およびディレクトリ領域に初期データを書き込む必要があります。これをシステムフォーマット（2次フォーマット）と呼んでいます。本機ではディスク管理する場合は、全てのディスクタイプとも必ずシステムフォーマットする必要があります。

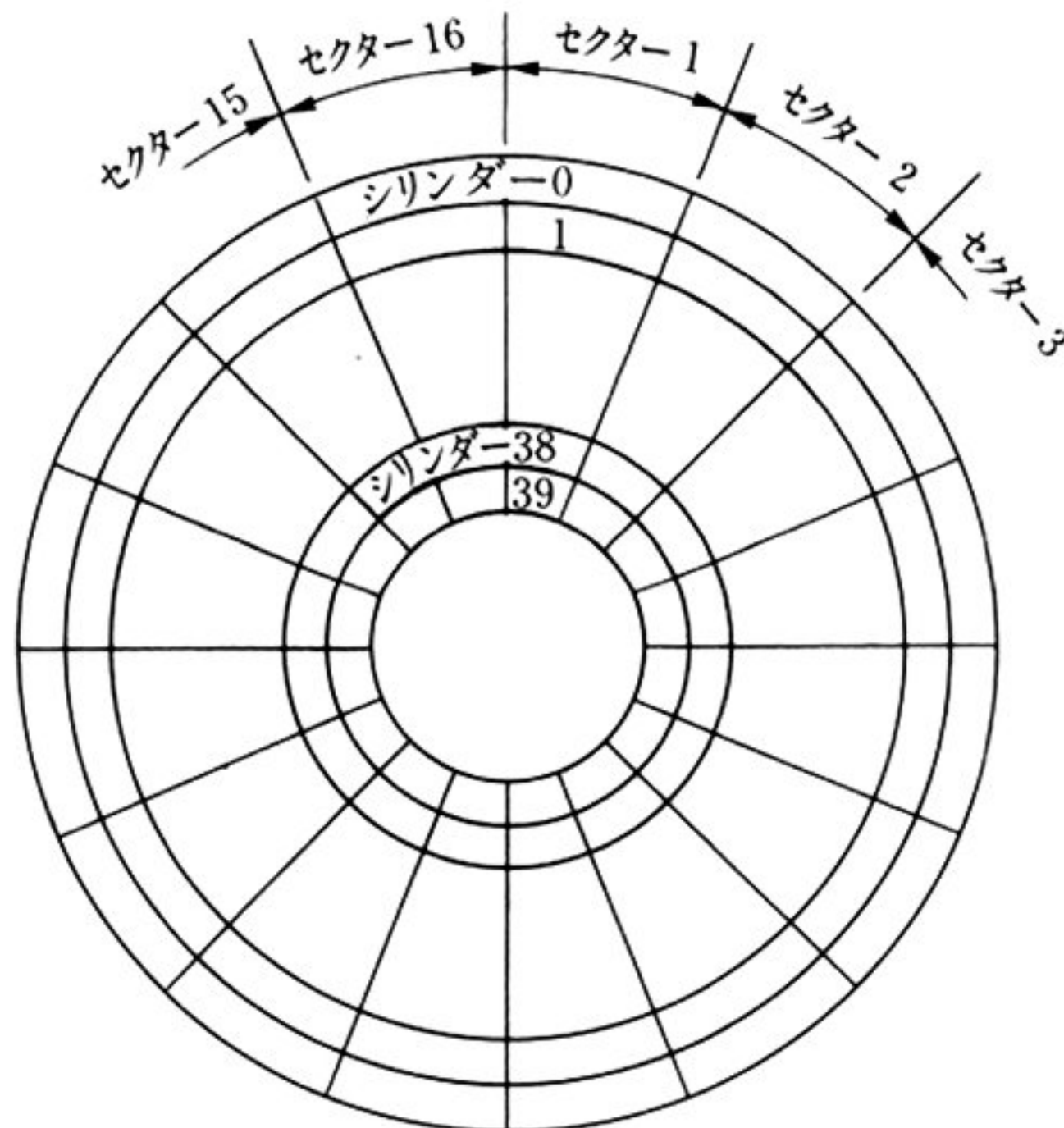
実際にディスクをフォーマットする場合は

ユーティリティプログラム中のフォーマットプログラムで実行してください。

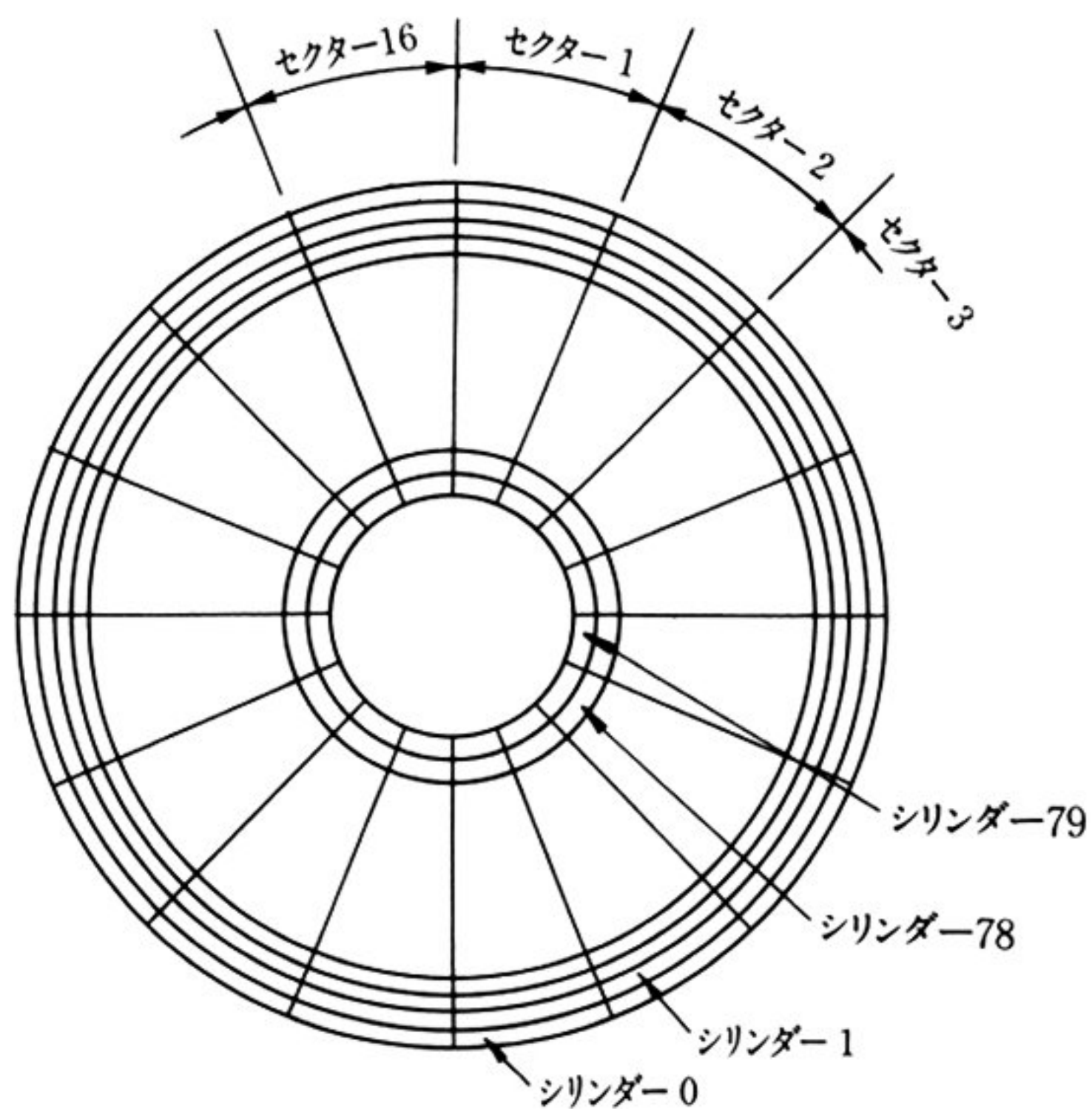
詳しくは『アプリケーションソフトの説明書』の「ディスクユーティリティ」をお読みください。  
このフォーマットプログラムでは、1次フォーマットと2次フォーマットを両方行ないますが、5  
インチFD・2HDおよび8インチFD・2D、1Sの市販ディスクは2次フォーマット（システ  
ムフォーマット）のみで使用可能です。

このシステムフォーマットを行なう場合は、INIT "デバイス名："を実行してください。

(『BASIC リファレンス マニュアル』2.4.1 INIT参照)

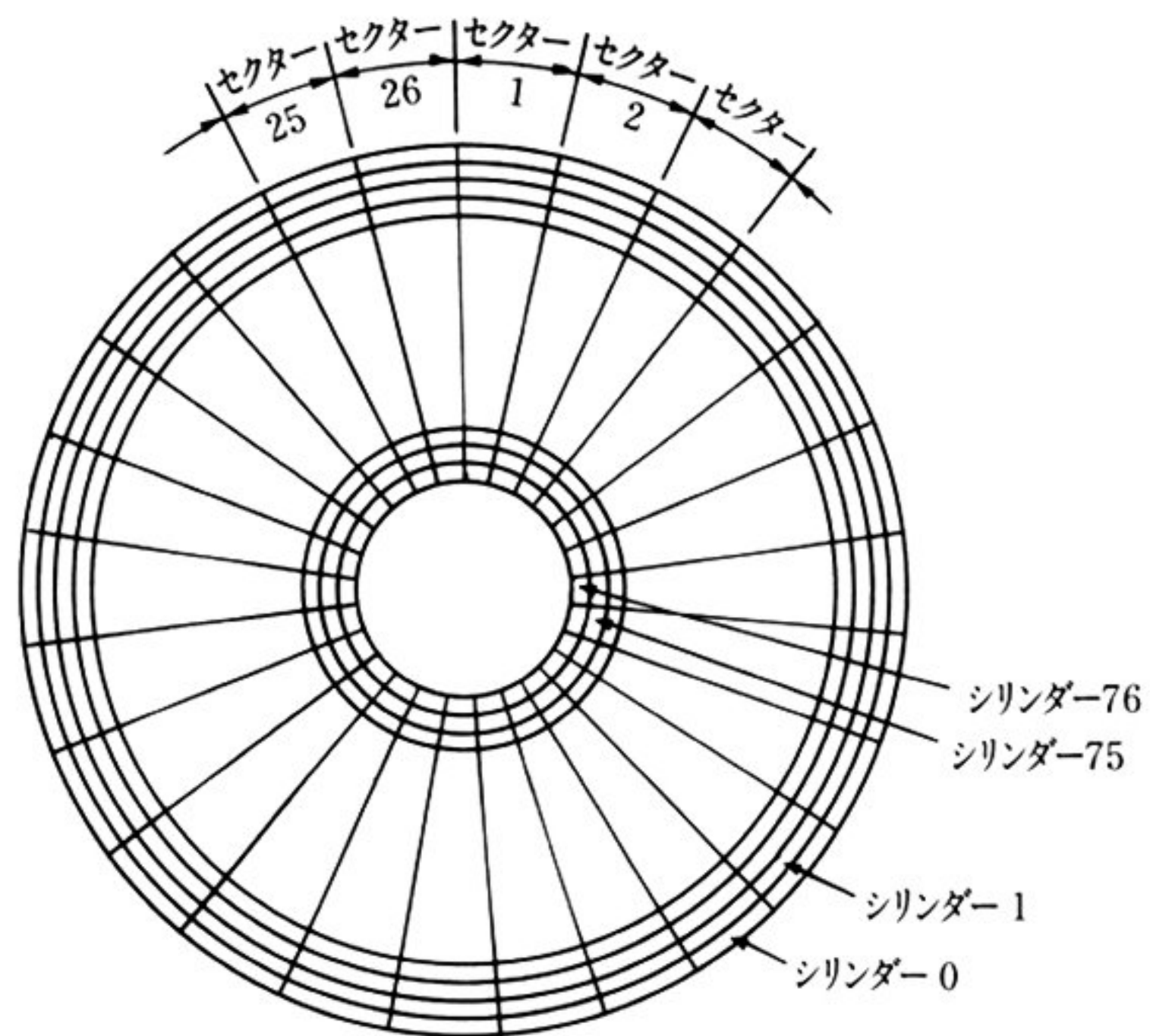


5インチFD 2Dタイプのディスク構造

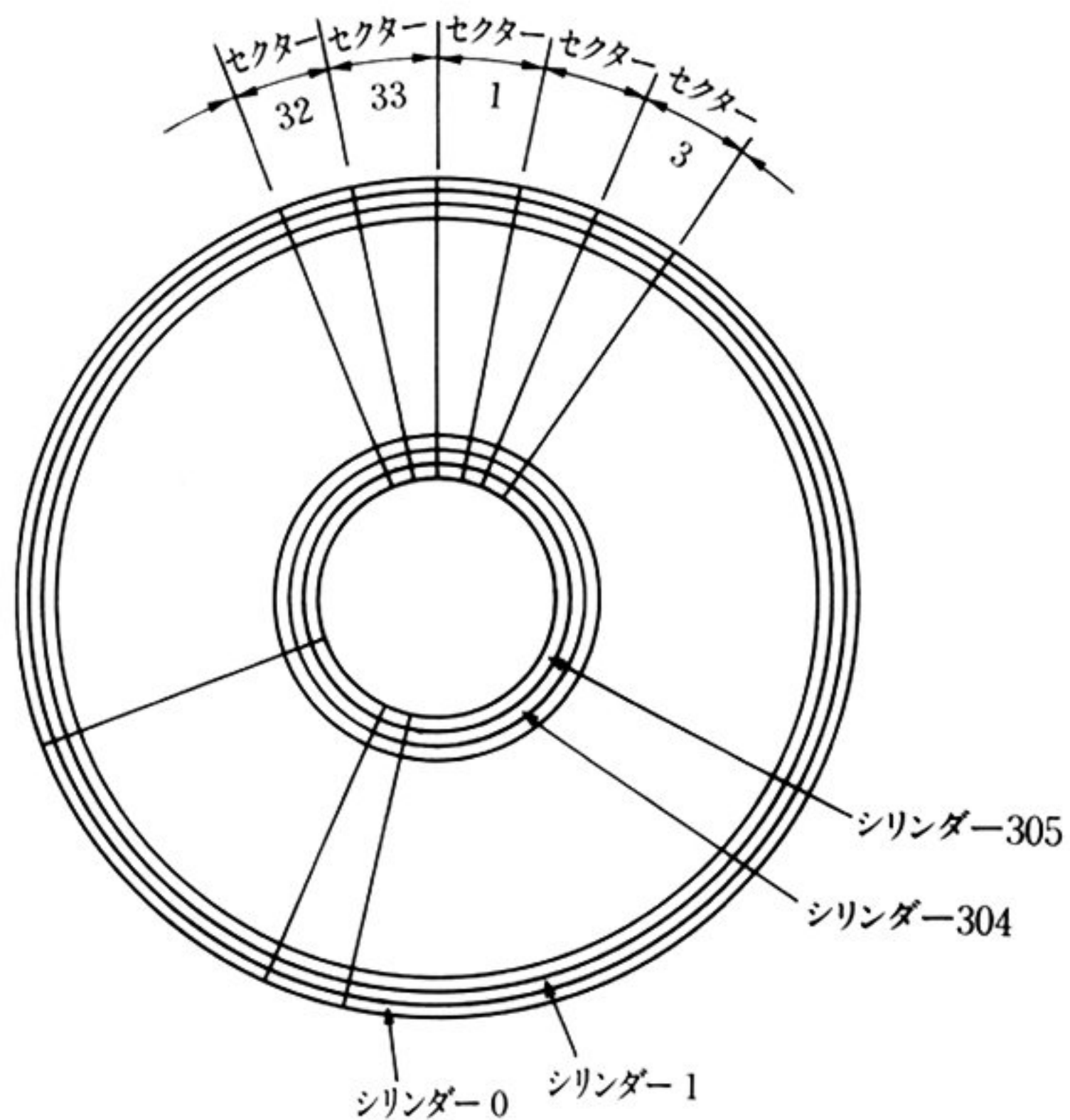


5インチFD 2DDタイプのディスク構造





5 インチ F D 2 H D      タイプのデヤスク構造  
 8 インチ F D 2 D 1 S



5 インチハードディスクのディスク構造

3インチまたは5インチ、8インチフロッピーディスクおよび、ハードディスクの全てのタイプのディスクへの書き込み／読み出しは、ヘッド番号、シリンダー番号、セクター番号から連続した論理アドレスに変換し、その最小単位のレコード（128バイト／256バイト）ごとに、可能ですが実際のファイル管理での最小単位は、16レコード＝1クラスタ（4Kバイト）で行なっています。

ディスク上におけるファイルの記録状態や使用状態の管理は、FAT（File Allocation Table）のクラスタ番号とディレクトリデータで行なっています。各ディスクタイプの記録容量によって FAT、ディレクトリ、データ領域は下記の構成となっています。

（単位＝レコード）

3インチまたは5インチフロッピーディスク				8インチフロッピーディスク	ハード ディスク
ディスク 項目	2D	2DD	2HD	2D	
システム領域	0	0	0	0	0
FAT領域	14	14 } 15	28 } 29	28 } 29	8 } 27
ディレクトリ 領域	16 } 31	16 } 31	32 } 47	32 } 47	48 } 63
データ領域	32 } 1279	32 } 2559	48 } 4003	48 } 4003	64 } 40127
代替領域	—	—	—	—	40128 } 40391

ハードディスクの使用終了時に、ヘッドをデータ領域以外に移動することにより、ディスクデータの破壊からの保護を行なっています。

8インチFDの片面単密度記録は 1レコード単位の書き込み／読み出しは可能であるがBASICでのファイル管理はサポートしていません。

### (1)ディレクトリ

ディレクトリはファイル管理のためのファイル名、格納先頭クラスタ番号、データ数、日付等を表わし、1ファイル32バイトで表現しています。

各タイプのディスクとも階層ディレクトリ形式なので、データ領域内に下位ディレクトリが設定され、各下位ディレクトリはディスクタイプに関係なく、1クラスタ分（4Kバイト：128ファ

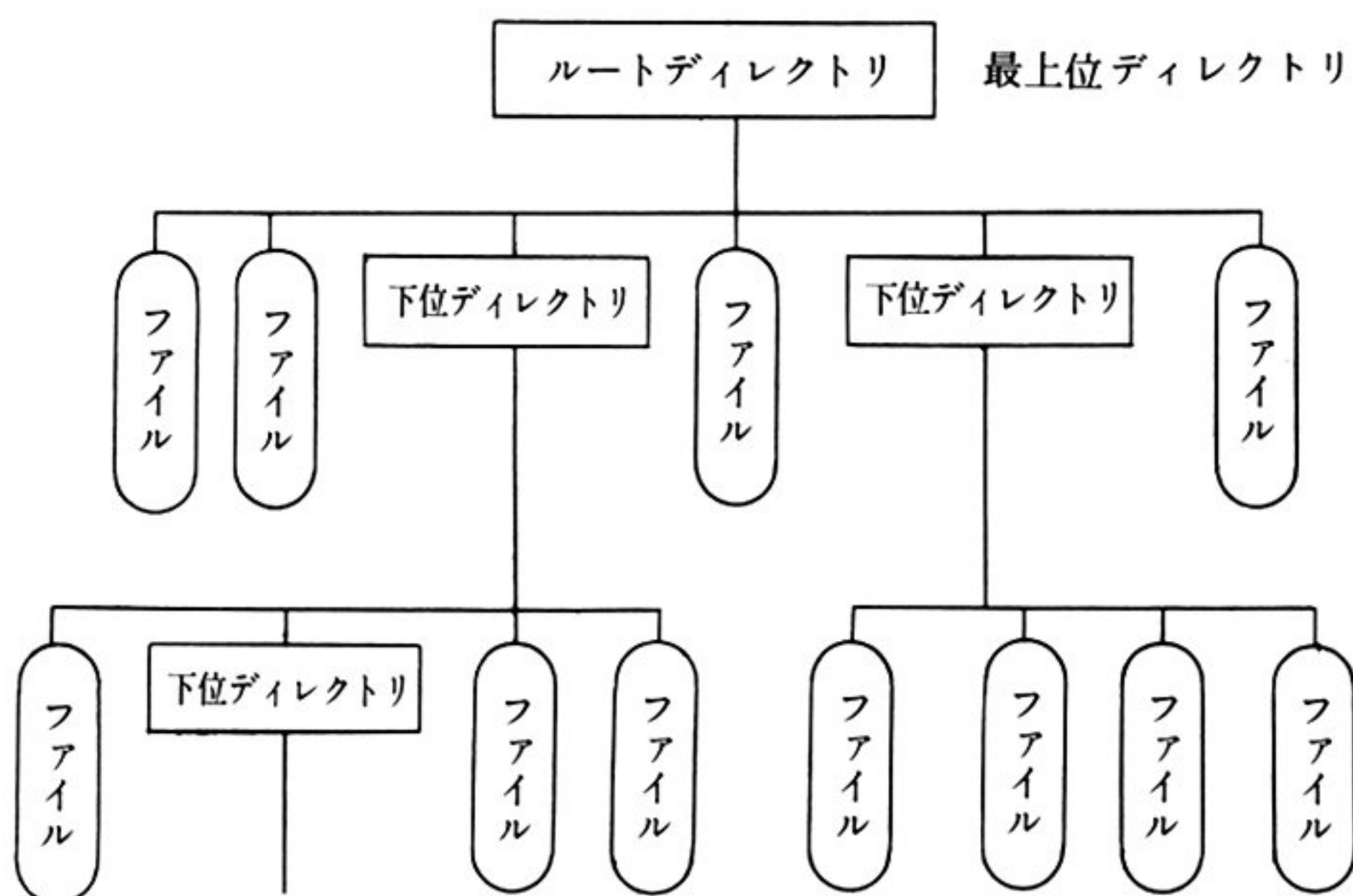


イル数)の領域を確保できます。最上位ディレクトリ(ルートディレクトリ)で管理できるファイル数はディスクタイプで異なり次表に示すとおりです。

ディスクタイプで  
管理可能な最大ファイル数

ディスク	項目	ルートディレクトリ 領域 レコード数	ファイル数
3インチ または 5インチ FD	2D	16	78
	2DD	16	158
	2HD	16	247
8インチFD	2D	16	247
ハード ディスク		16	2504

ディレクトリの構成は下図の様なツリー構造になります。



ディレクトリのファイルに対する構成を次に示します。(1 ファイル=32 バイト)

### 1 ファイル名に対するディレクトリの構成

	内 容
1 バイト目	<p>種類を表わす。 00 は KILL されたファイルまたは未使用領域。FF は使用ディレクトリテーブルの終り。</p> <p>bit 0 が 1 …… Bin ファイル (機械語で書かれたファイル)</p> <p>bit 1 が 1 …… Bas ファイル (BASIC テキストで書かれたファイル)</p> <p>bit 2 が 1 …… Asc ファイル (ASCII セーブされたファイル)</p> <p>bit 4 が 1 …… FILES で表示しない: 0 …表示する</p> <p>bit 5 が 1 …リードアフターライト ON: 0 …OFF</p> <p>bit 6 が 1 …書き込み禁止ファイル: 0 …書き込み OK</p> <p>bit 7 が 1 …下位ディレクトリ</p> <p>bit 3 は予備</p>
2 バイト目～14 バイト目	ファイル名 (13 文字)
15 バイト目～17 バイト目	ユーザー指定 エクステンションエリア (3 文字)
18 バイト目	パスワードのバック (無指定なら 20 (16) の値)
19・20 バイト目	ファイルのバイト数 (Bas および Obj のみ有効)
21・22 バイト目	ファイルのメインメモリー先頭アドレス (Obj のみ有効)
23・24 バイト目	ファイルのメインメモリー実行アドレス (Obj のみ有効)
25 バイト目～29 バイト目	<p>作成された年、月、曜日、日、時、分が書き込まれています。</p> <p>例: '84 年 12 月 01 日 土曜日、16 時 36 分</p> <p>先頭から、年 年 月 曜 日 日 時 時 分 分</p> <p>84 C 6 01 16 36</p>
30 バイト目～32 バイト目	<p>ファイル先頭 クラスタ値</p> <p>30 バイト目 HIGH バイト</p> <p>31 バイト目 LOW バイト</p> <p>32 バイト目 MIDDLE バイト</p>

### (2) F A T

F A T はディスクで管理されるファイルのチェーン状態を表わし、2 バイトのクラスタ番号で表現しています。F A T の必要容量 (バイト数) は (管理可能な最大ファイル数) × 2 バイトです。



F A T構成としては

アドレス

0 } 7F	0 ~ 7F 80 ~ 8F	下位バイト
80 } FF	0 ~ 13	上位バイト

図の様にF A T領域の1レコードを128バイトに分割し、各々をクラスタの下位、上位バイトに指定しています。

各ディスクタイプによってディスク管理用の予約バイトとして F A T領域の先頭から

3インチまたは5インチFD 2D, 2DD	2バイト
5インチFD 2HD 8インチFD 2D	3バイト
ハードディスク	4バイト

上表のように必要です。F A Tデータとしては、次のようなものがあります。

ディスク 項目	3インチまたは5インチFD			8インチFD	ハード ディスク
	2D	2DD	2HD	2D	
ファイルがチェ ーンしている クラスタ	02 ~ 4F	02 ~ 7F 100 ~ 11F	03 ~ 7F 100 ~ 179	03 ~ 7F 100 ~ 179	04 ~ 7F 100 ~ 17F 1300 ~ 1353
ファイル終了 クラスタ	80 ~ 8F	80 ~ 8F	80 ~ 8F	80 ~ 8F	80 ~ 8F
未使用クラスタ	0	0	0	0	0

# 13章

## RS-232Cインターフェイスの使い方

本機はRS-232Cインターフェイスを内蔵しており、RS-232Cインターフェイスを持っている機器との間でシリアルデータの送・受信ができます。

### 1

### 接続方法

#### (1) 接続例

RS-232Cインターフェイス機能を使用して、以下に示すようなシステムのもとで、RS-232Cインターフェイス機能を持つ他のコンピュータあるいは周辺機器との間でデータの送・受信ができます。

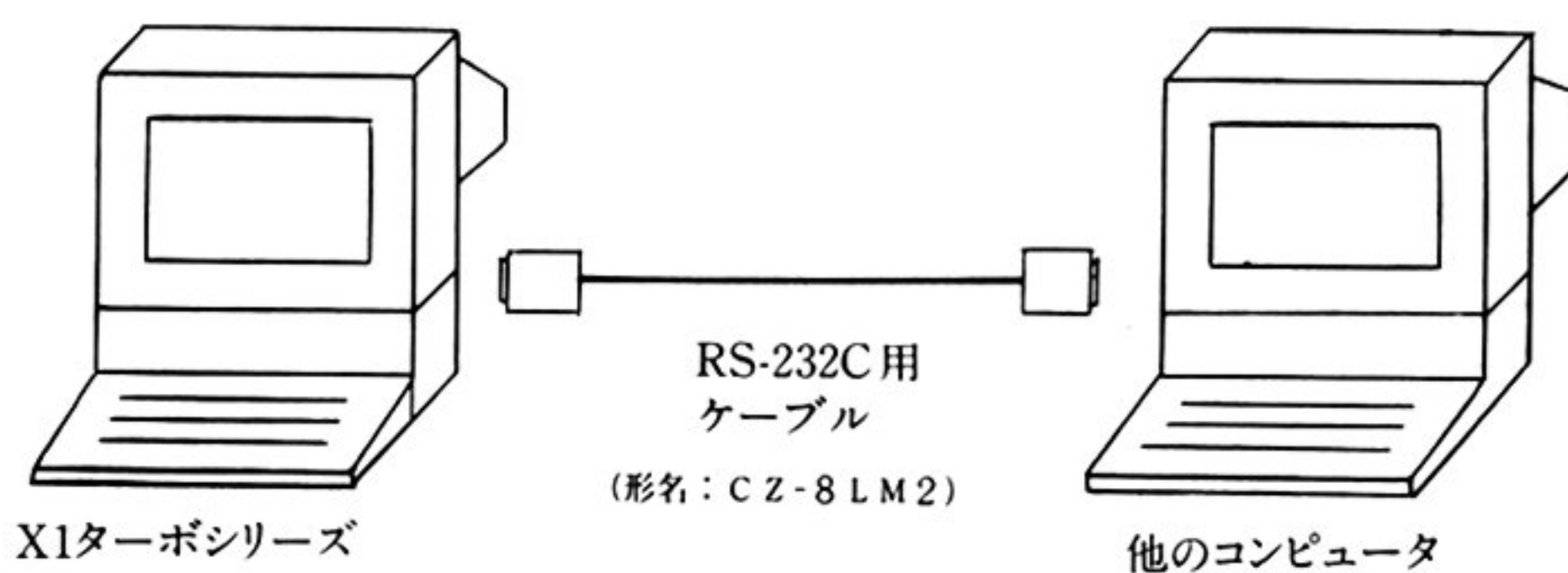
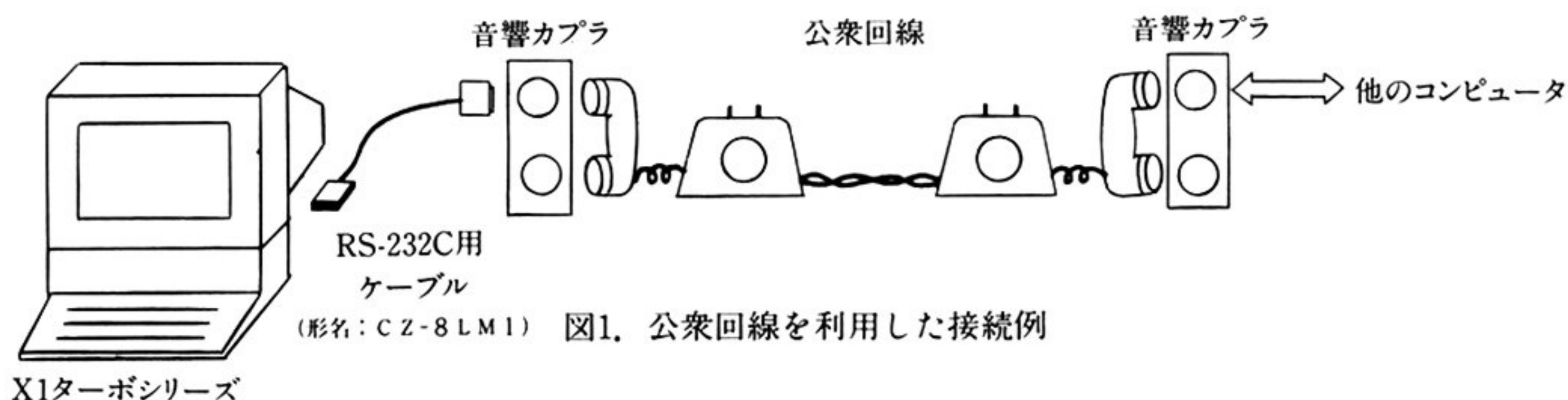


図2. 他のコンピュータとの接続

#### (2) RS-232Cインターフェイス信号端子表

RS-232Cインターフェイス用コネクタには25ピンD-subコネクタのメス形を使用しており、そのピン配置と信号端子は次のとおりです。

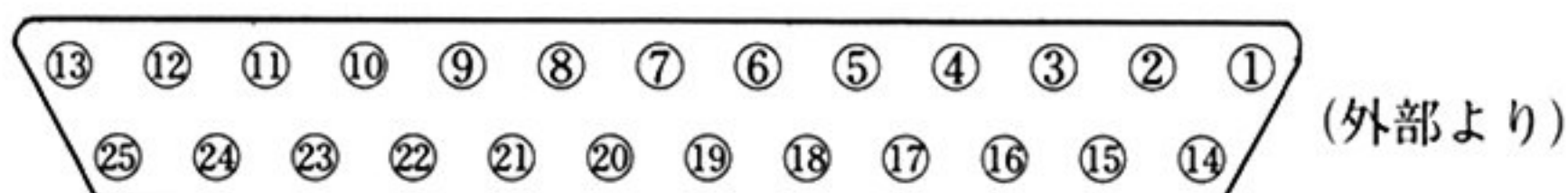


図3. RS-232Cインターフェイスコネクタピン配置



RS-232Cコネクタ端子表

端子番号	信号名	信号方向	機能
1	FG (保安用アース)	↔	保安用アース
2	TxD (送信データ)	→	送信データ
3	RxD (受信データ)	←	受信データ
4	RTS (送信要求)	→	ONにより相手を送信可能とする
5	CTS (送信可)	←	ONの時送信可能と判断する
6	DSR (データセットレディ)	←	ONの時相手が送・受信の準備ができていると判断する
7	SG (信号用アース)	↔	信号用アース
8	CD (キャリア検出)	←	ONの時受信データ有効と判断する
9～14	NC		
15	ST2 (送信信号エレメントタイミング)	←	同期式通信時の送信信号エレメントタイミング信号を入力する
16	NC		
17	RT (受信信号エレメントタイミング)	←	同期式通信時の受信信号エレメントタイミング信号を入力する
18,19	NC		
20	DTOR (データターミナルレディ)	→	ONにより送・受信の準備ができたことを知らせる
21	NC		
22	CI (被呼表示)	←	ONにより相手から呼び出されていると判断する
23	NC		
24	ST1 (送信信号エレメントタイミング)	→	同期式通信時の送信信号エレメントタイミングを出力する
25	NC		

→OUT

←IN

### (3) 接続方法

相手機器との接続には両端に25ピンD-subコネクタのオスを持つRS-232C用ケーブルを使用します。相手機器がモデム（音響カプラ含む）かコンピュータかによりコネクタ信号の方

向が異なります。相手機器によっては使用コネクタまたは信号配置が異なるものもあるため、相手機器の説明書を熟読の上接続してください。以下に一般的な接続例を示します。

図4 モデムや音響カプラと接続する場合のケーブル配線例

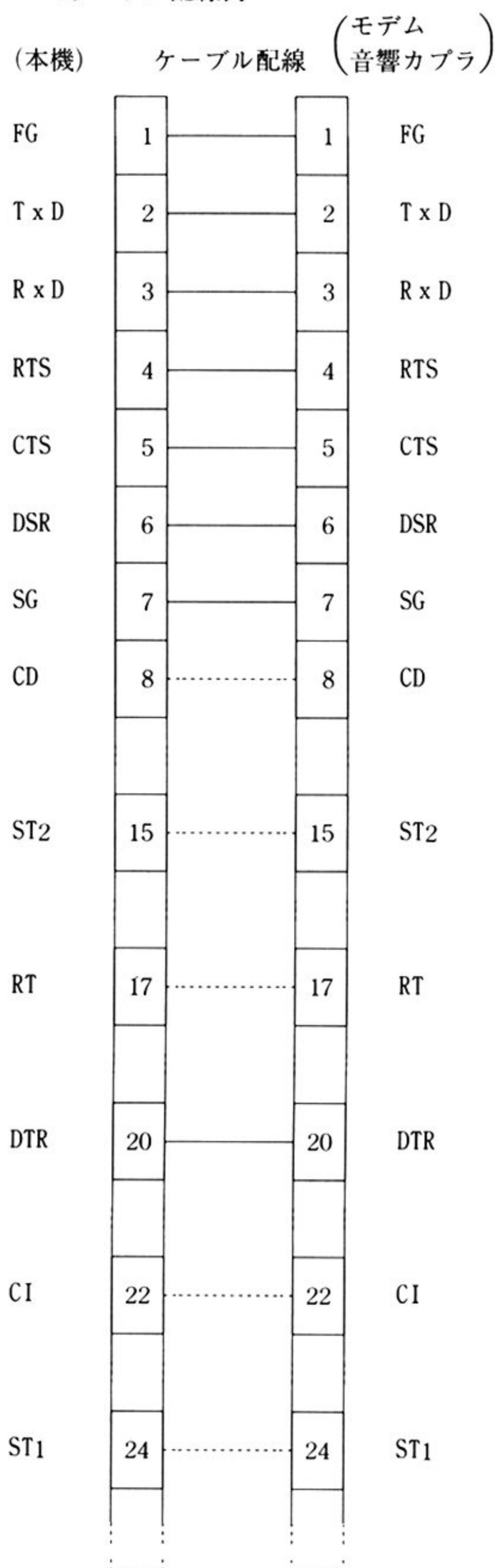
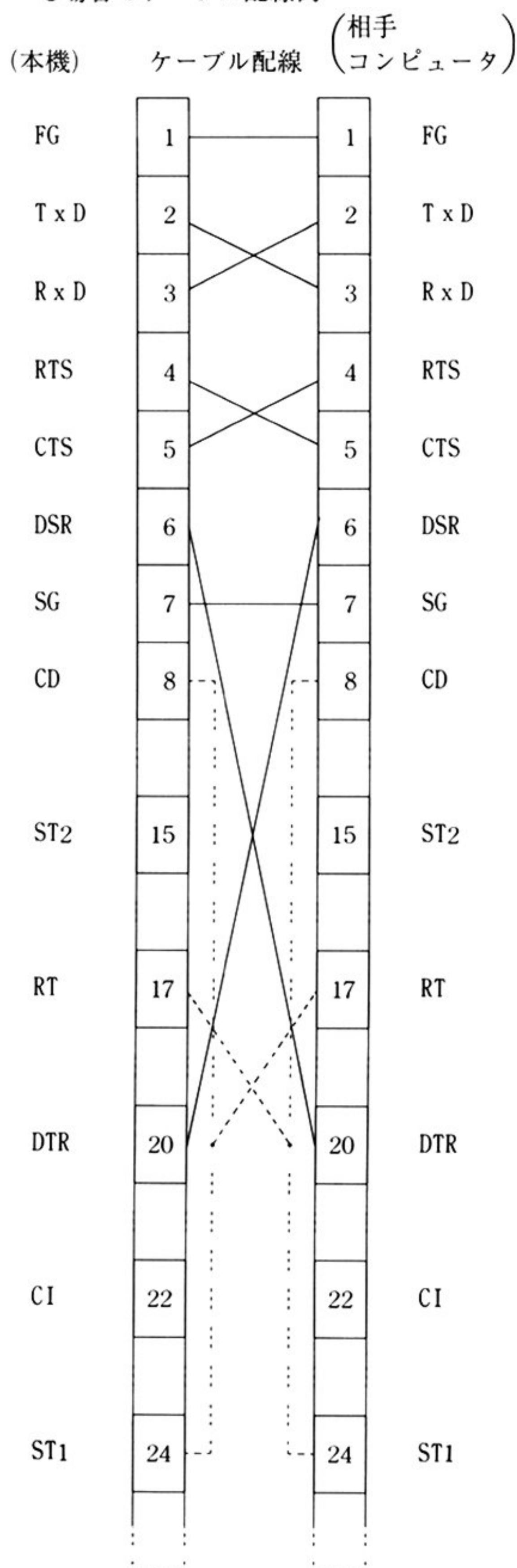


図5 他のコンピュータ（本機を含む）と接続する場合のケーブル配線例



…で示す信号線はBASICではサポートしておらず特に接続する必要はありません。外部同期を使用する場合など、ユーザーがマシン語レベルでソフトを作成して使用する時に必要に応じて接続します。



## 2

## RS-232Cインターフェイスのデータ信号フォーマット

RS-232Cインターフェイスは、コンピュータからの8ビットパラレルデータをシリアルデータに変換してRS-232Cで規定された電圧レベル（±12V）に変換して送信したり、逆に受信したシリアルデータをパラレルデータに変換するものです（図6参照）。シリアルデータの形式は同期式、非同期式通信方法によって異なります。本機のBASICでは非同期通信方式をサポートしています。この場合のデータ信号フォーマットを図7に示します。機器間でデータの送・受信を行なう場合にはこのデータ信号フォーマットおよびボーレート（信号の伝送速度）を一致させる必要があります。設定方法については次項のBASIC上でのRS-232Cインターフェイス取扱方法で説明します。

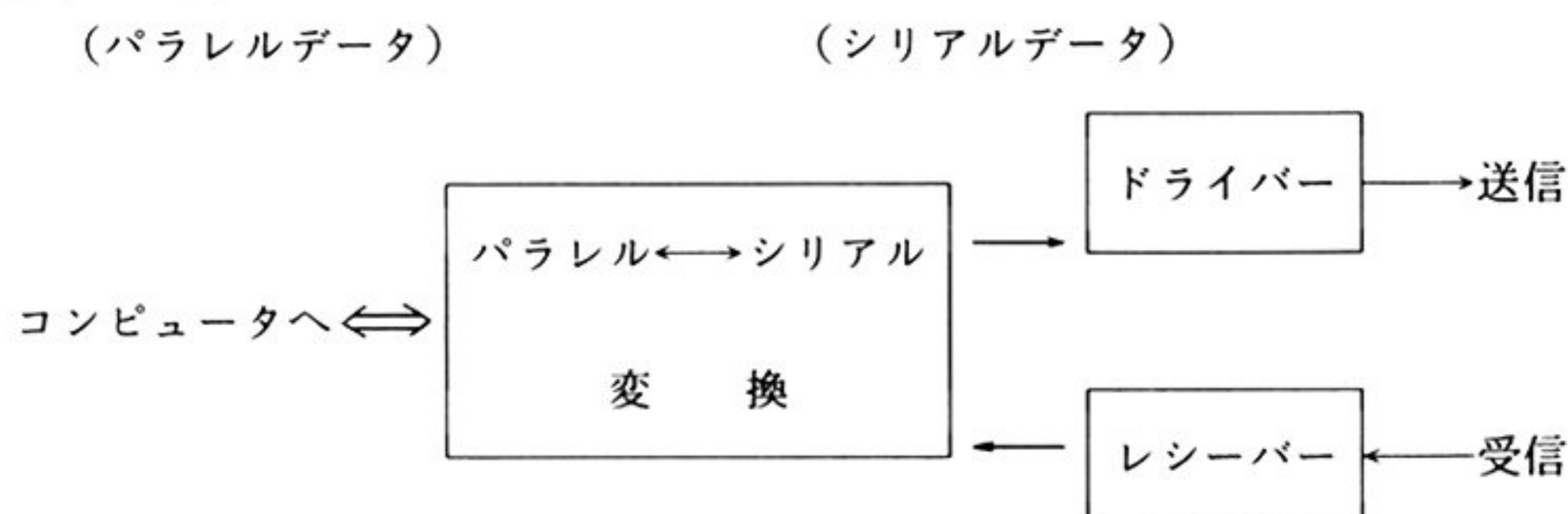


図6 RS-232C概念図

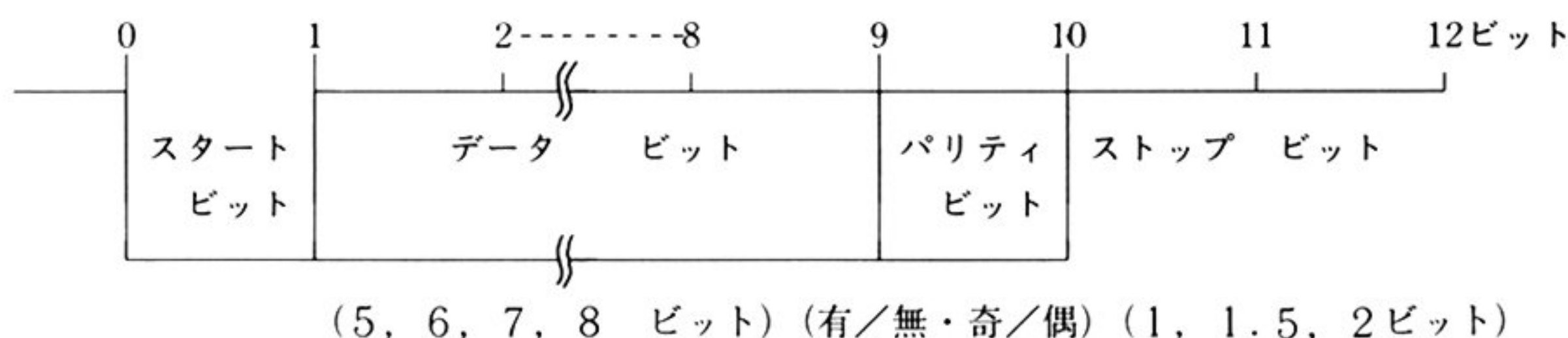


図7 シリアルデータ フォーマット

## 3

## BASICでRS-232Cインターフェイスを取り扱う方法

RS-232Cインターフェイスで扱うデータはファイルという概念で取り扱い、BASICの入出力命令を使って、一般の入出力装置として使用します。RS-232Cインターフェイスのファイルディスクリプタは次の形式で表わされます。

"COM:通信パラメータ"

通信パラメータは次のとおりです。

〈ボーレート〉 〈パリティ〉 〈データビット長〉 〈ストップビット長〉 〈通信制御指定〉 〈カナの表現方法指定〉 〈CR、LFコードの送信処理〉 〈CR、LFコードの受信処理〉 〈日本語文字列の表現方法指定〉 〈エンドコード指定〉

〈ボーレート〉

0～6の数値によりボーレートを設定します。

数 値	0	1	2	3	4	5	6
ボーレート(ボー)	1 5 0	3 0 0	6 0 0	1 2 0 0	2 4 0 0	4 8 0 0	9 6 0 0

#### 〈パリティ〉

E ……偶数パリティ・チェックを使用します。

O ……奇数パリティ・チェックを使用します。

N ……パリティ・チェックを使用しません。

#### 〈データビット長〉

5 ……データビット長を5ビットとします。

6 ……データビット長を6ビットとします。

7 ……データビット長を7ビットとします。

8 ……データビット長を8ビットとします。

#### 〈ストップビット長〉

1 ……ストップビット長を1ビットとします。

2 ……ストップビット長を1.5ビットとします。

3 ……ストップビット長を2ビットとします。

#### 〈通信制御指定〉

通信制御方法を選択します。

X ……X O N / X O F Fコードによる制御を行ないます。

R ……R T S制御信号のO N / O F Fによる制御を行ないます。

$\left\{ \begin{array}{l} \text{N} \\ \text{省略} \end{array} \right\}$  ……どちらの制御も行ないません。

この通信制御とは、データ送受信時に受信側においてデータの受信処理が間に合わないときに、送信側に対して送信の一時停止を要求し、受信できる状態になると送信再開を要求することです。Xを指定するとデータとしてX O F Fコード(&H 1 3)を送信側へ送り返すことにより送信の一時停止を要求し、X O Nコード(&H 1 1)により送信再開を要求します。また、Rを指定すると、R T S制御信号線をO F Fにすることにより送信の一時停止を要求し、O Nにすることにより送信再開を要求します。

#### 〈カナの表現方法指定〉

S ……データ7ビットモードでカナの送受信ができます。

$\left\{ \begin{array}{l} \text{N} \\ \text{省略} \end{array} \right\}$  ……データ7ビットモードでカナの送受信ができません。

データ7ビットモードでカナの送受信は、シフトアウトコードS O (&H 0 E)があるとそれ以後のデータはカナとみなし、シフトインコードS I (&H 0 F)があるとそれ以後のデータを英数字とみなします。

#### 〈C R, L Fコードの送信処理〉

C ……一連の文字列送信後、C Rコード(&H 0 D)を復帰+改行コードとして処理します。

$\left\{ \begin{array}{l} \text{L} \\ \text{省略} \end{array} \right\}$  ……一連の文字列送信後、C Rコード(&H 0 D) + L Fコード(&H 0 A)を復帰+改行コードとして送信します。



## ＜CR LFコードの受信処理＞

C……………1つのCRコード（&H0D）の受信で復帰+改行コードとして処理します。

{ L } ……CRコード（&H0D）のみ受信すると、それはデータとして処理され、CRコード  
 { 省略 } （&H0D）とLFコード（&H0A）を連続して受信すると復帰+改行コードとして処理します。

## ＜日本語文字列の表現方法指定＞

日本語文字列の表現方法を選択します。

J……………漢字インコードKI（&H1B4B）で日本語文字列の始まりを示し、漢字アウトコードKO（&H1B48）で日本語文字列の終わりを示します。

{ N } ……シフトJIS漢字コードを使用します。  
 { 省略 }

## ＜エンドコード指定＞

データ転送の終了を判断するためのエンドコードを指定します。エンドコードを設定するとSAVE命令を実行した場合プログラムデータの送信後指定されたエンドコードを送信してSAVE動作を終了します。LOAD命令を実行した場合、指定されたエンドコードを受信した時点でLOAD動作を終了します。RS-232CファイルをOPEN命令によりアウトプットオープンした場合、CLOSE命令の実行によりエンドコードが送信されます。また、インプットオープンした場合、EOF関数はエンドコードを受信すると真の値となります。

このエンドコードとしてコントロールコード（&H00～&H1F）の中から任意の1つを選択できます。パラメータ値としてはキャラクタコード（&H40～&H5F）に対応する文字を使用します（&H60～&H7Fも可）。たとえば、D（またはd）と指定すると、エンドコードとしてCTRL-D（&H04）が使用されます。またこのパラメータを省略するとエンドコードの送受信処理は行ないません。

**（注意）** 通信パラメータの〈ボーレート〉から〈ストップビット長〉までは省略できませんが、それ以降のパラメータは省略できます。ただし、途中のパラメータを省略して次のパラメータを指定することはできません。

日本語文字の送・受信はデータ長8ビットのときに可能です。ただし、7ビットモードでもSI/SOコード制御及びKI/KOコード制御を使用することにより可能です。この場合のデータ形式は次のとおりです。

SI	KI	漢字	KO	SO	カナ	SI	KI	漢字	KO	SO	
----	----	----	----	----	----	----	----	----	----	----	--

送／受信方向 →

また、KI/KOコード制御にて日本語文字を送・受信する場合は、漢字表示モードKMODE1に設定してください。

## 4.1 RS-232Cインターフェイスに関する入出力命令

**SAVE "COM:通信パラメータ"**

メインメモリ上のBASICプログラムをアスキー形式にて、RS-232Cインターフェイスを介して相手機器へ送信します。

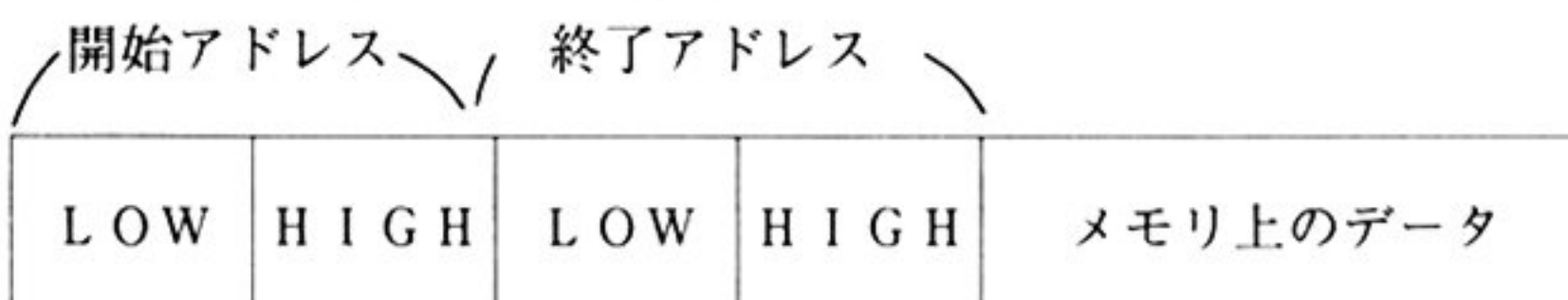
**LOAD "COM:通信パラメータ"**

相手機器より送られてきたアスキー形式のBASICプログラムをRS-232Cインターフェイスを介してメインメモリへ読み込みます。

**SAVEM "COM:通信パラメータ", <開始アドレス>, <終了アドレス>**

メインメモリ上のマシン語のプログラムをRS-232Cインターフェイスを介して相手機器へ送信します。

この場合の送信データ形式は次のとおりです。

**LOADM "COM:通信パラメータ"**

相手機器より送られてきたマシン語プログラムをRS-232Cインターフェイスを介してメインメモリへ読み込みます。

この場合送られてきたデータはSAVEM命令の項に示すデータ形式になっている必要があります。

**OPEN " {<sup>I</sup>O<sub>C</sub> } ", [#] ファイル番号, "COM:通信パラメータ"**

I .....インプットオープン (受信用)

O .....アウトプットオープン (送信用)

C .....コミュニケーションオープン (送・受信用)

RS-232Cファイルを開き (通信パラメータの設定を行なうこと)、PRINT #、INPUT # 命令等でデータの送・受信ができる状態にします。コミュニケーションオープンはRS-232Cファイルについてのみ有効です。コミュニケーションオープンすると同一ファイル番号でデータの送受信ができます。

**PRINT # ファイル番号, [X<sub>1</sub>, X<sub>2</sub>, .....]**

X<sub>1</sub>, X<sub>2</sub> .....出力する式または文字列

RS-232Cファイルがアウトプットオープンされている場合、式または文字列を相手機器へ送信します。文字列送信後、通信パラメータの<CR、LFコードの送信処理>でCが設定されている場合にはCRコードがまたLが設定されている場合CR+LFコードが送信されます。

**INPUT # ファイル番号, X<sub>1</sub> [, X<sub>2</sub>, .....]**

X<sub>1</sub>, X<sub>2</sub> .....入力したデータを入れる変数

RS-232Cファイルがインプットオープンされている場合、相手機器より受信したデータを変数に読み込みます。



**INPUT#** ファイル番号, 文字変数

**LINE INPUT#** ファイル番号, 文字変数

文字変数………入力したデータを入れる文字型変数

RS-232Cファイルがインプットオープンされている場合、相手機器より受信したデータ(255文字まで)を文字変数に読み込みます。通信パラメータの〈CR、LFコードの受信処理〉でCが設定されているとCRコードが、またLが設定されているとCR+LFコードがくるとINPUT動作を完了します。

**WRITE#** ファイル番号, [X<sub>1</sub>, X<sub>2</sub>, ……]

PRINT#文と同様ですが、データの区切りにはカンマ(,)を送信し、文字列データのときはダブルクォーテーション(")をつけて、詰めて送信します。

**INPUT\$(n, [#] ファイル番号)**

n………読み込む文字数

RS-232Cファイルがインプットオープンされている場合、相手機器より受信したn個の文字がこの関数の値になります。

**LOC (ファイル番号)**

受信バッファ(64バイト)にたまっている文字数がこの関数の値となります。

**EOF (ファイル番号)**

受信動作開始時は偽の値(0)がこの関数の値となり、相手機器よりエンドコードが送られてくると真の値(-1)がこの関数の値となります。

**CLOSE** [[#] ファイル番号1, [#] ファイル番号2, ……]

OPENによって開かれたファイルを閉じます。RS-232Cによる送・受信を終了します。アウトプットオープンされていた場合にはエンドコードを送信します。

**DEVICE "COM:"**

デバイス名を省略をしたときに用いられるデバイス名を"COM:"に設定します。LOAD、SAVE等のコマンドにおいてデバイス名を省略すると、デバイス名は"COM:"と判断します。

(使用例) DEVICE "COM:"

SAVE "6N83XNLLJD"

の命令を実行するとRS-232Cポートへプログラムを送出します。

## 4.2 割り込み処理

INPUT#、INPUT\$命令を実行すると、受信データ待ち状態となり、データを受信するまでプログラムは停止します。ON COM GOSUB命令を使用することにより、この受信待ち時間には他の仕事をさせておき、データを受信した時点で指定された行番号からの処理ルーチンの実行を開始することができます。この処理ルーチンからの復帰はRETURN命令によって行なわれます。

ON COM GOSUB { 行番号  
                  "ラベル名" }

RETURN { { 行番号  
              "ラベル名" } }

行番号、ラベル名を省略すると、中断した所から処理を再開します。

また、COM ON/OFF/STOP 命令により RS-232C からの割り込みの許可、禁止、停止を設定できます。

COM ON .....割り込み許可  
COM OFF .....割り込み禁止  
COM STOP .....割り込み一時保留

#### サンプル プログラム

ディスクにあるアスキー形式のプログラムを読み込み RS-232C インターフェイスより送信します。

```
10 INPUT "File name:",F$
20 F$=LEFT$(F$+SPACE$(13),13)
30 OPEN "I",#1,"1:"+F$
40 OPEN "O",#2,"COM:6N83XNCCND"
50 IF EOF(1)=-1 THEN 100
60 LINPUT #1,D$
70 PRINT D$
80 PRINT#2,D$
90 GOTO 50
100 CLOSE #1,#2
110 END
```

RS-232C インターフェイスより受信したプログラムをグラフィックメモリに書き込みます。

```
10 INPUT "File name:",F$
20 F$=LEFT$(F$+SPACE$(13),13)
30 OPTIONSCREEN 4
40 INIT"MEM:"
50 OPEN "O",#1,"MEM:"+F$
60 OPEN "I",#2,"COM:6N83XNCCND"
70 IF EOF(2)=-1 THEN 120
80 LINPUT #2,D$
90 PRINT D$
100 PRINT#1,D$
110 GOTO 70
120 CLOSE #1,#2
130 END
```

ライン命令実行中に RS-232C インターフェイスよりデータを受信すると割り込みがかかり、そのデータを画面に表示して前の処理を再開します。

```
10 INIT:CLS4
20 OPEN "I",#1,"COM:6N83XN"
30 ON COM GOSUB 100
40 COM ON
50 FOR I=0 TO 99
60 LINE(0,100+I)-(319,199),PSET,I,BF
70 NEXT
80 COM OFF:CLOSE #1
90 END
100 PRINT INPUT$(LOC(1),1);
110 RETURN
```



図8にRS-232C周辺のシステム図を示します。データ伝送におけるボーレート（伝送速度）を決定する基本クロックの作成には、CTC（カウンタ／タイマー用LSI）を使用し、シリアルデータ通信用LSIにはSIOを使用しています。RS-232Cインターフェイスを使用してデータ通信を行なう場合にはこのCTCとSIOを通信形体に合わせて設定します。以下、CTC並びにSIOの設定方法について説明します。

#### (1) CTC (Z-80A CTC) の設定

CTCはカウンタ／タイマー機能を持ち、内部に4個の独立したチャンネル（チャンネル0～3）を持っています。このうちチャンネル1と2の出力をSIOに入力し、データ通信におけるボーレートを決定する基本クロックとしています。（ただし、チャンネル2の出力はマウス用に使用しているのでRS-232Cインターフェイスで使用するのはチャンネル1の出力のみです）CTCの各チャンネルのI/Oアドレスは次のとおりです。

チャンネル0.....1FA0（16進数）  
 チャンネル1.....1FA1  
 チャンネル2.....1FA2  
 チャンネル3.....1FA3

CTCはカウンタまたはタイマーモードとして使用でき、モードにより分周機能が異なります。ボーレート決定用基本クロックを作るにはカウンタモードで使い、2MHz基本クロックを1/1～1/256まで分周できます。

CTCを動作させるにはまず、チャンネル制御レジスタにそのチャンネルの使用モード等を決定する制御語（ここでは45（16進数））を設定し、つづいて時間定数レジスタに分周比として0～FF（16進数）を設定します。

```
LD BC, 1FA1H
LD A, 45H
OUT (C), A
LD A, 0DH
OUT (C), A
```

これによりCTCのチャンネル1がカウントを開始し、ボーレート決定用基本クロックをSIOへ出力します。ボーレートと時間定数レジスタ値とは一義的には定まらずSIOの設定により変わります。これについて次のSIOの項で説明します。

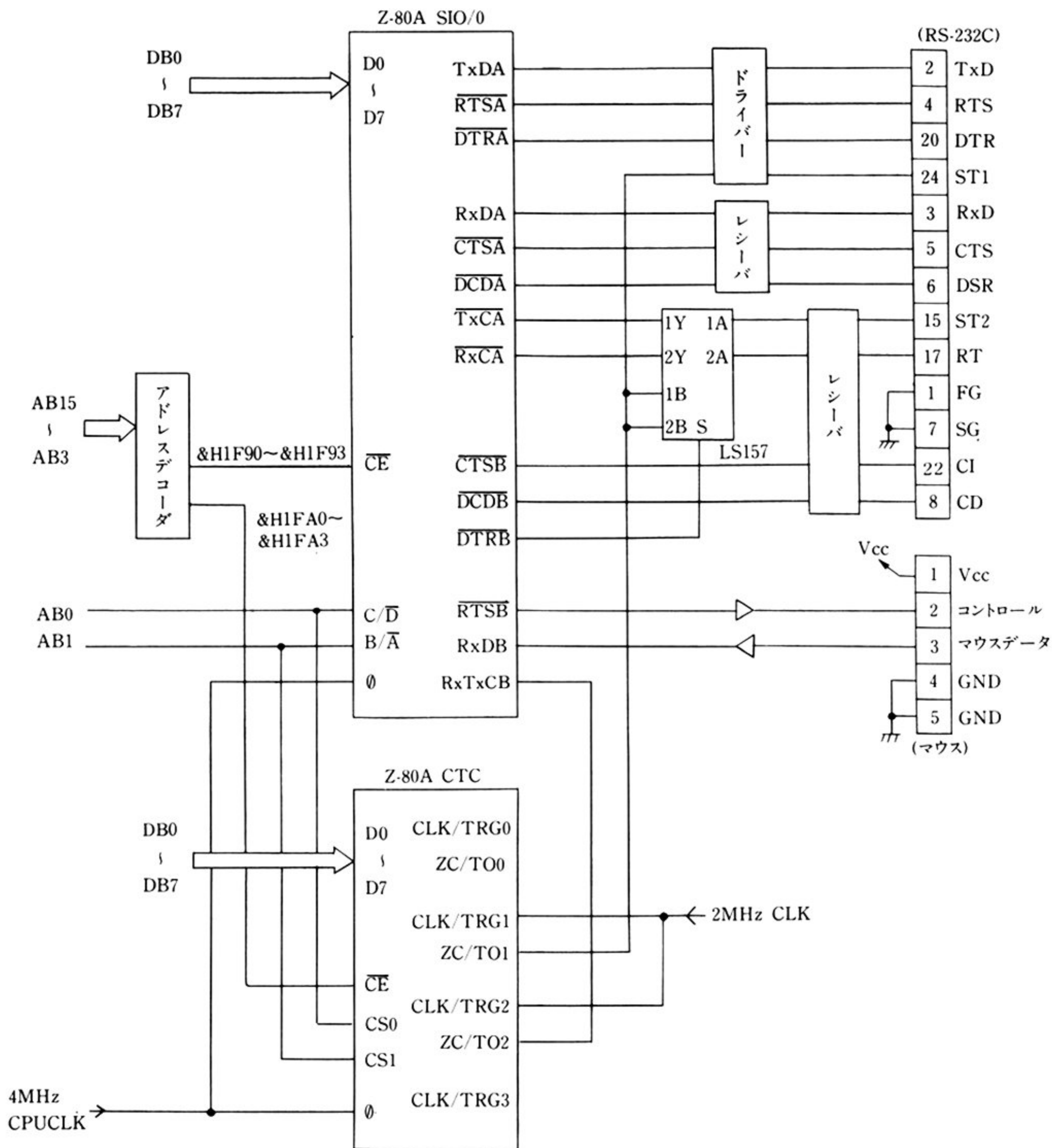


図 8. RS-232C 周辺回路図

## (2) SIO (Z-80A SIO/0) の設定

SIO はシリアルデータ通信用チャンネルを 2 つ (チャンネル A、B) 持っており、チャンネル A を RS-232C インターフェイス用に、またチャンネル B をマウス制御用に使用しています。SIO を動作させるにはまずコマンド書き込みにより通信モードを設定し、SIO を初期設定した後データの送・受信を行ないます。SIO のチャンネル、コマンド/データに対する I/O アドレスは次のとおりです。



チャンネルAのデータ………1F90 (16進数)

チャンネルAのコマンド………1F91

チャンネルBのデータ………1F92

チャンネルBのコマンド………1F93

SIOにはコマンドの書き込みレジスタを7つ持っています。これらのレジスタにコマンドを書き込む場合には、まず書き込みレジスタ0により実際に書き込みたいレジスタの番号を指定します。これによりそのレジスタへの書き込みが可能となりコマンドを書き込みます。たとえば書き込みレジスタ4にコマンドを書き込む場合は

LD A, 04H

LD BC, 1F91H

OUT (C), A

LD A, data……… (コマンドデータ)

OUT (C), A

のようにします。この操作を必要とする各レジスタについて行なうことによりSIOは初期化されデータの送・受信が可能となります。データを送・受信する場合には次のようにします。

(送信) LD A, data……… (送信データ)

LD BC, 1F90

OUT (C), A

(受信) LD BC, 1F90

IN A, (C) ……… (アキュムレータに受信データが入る)

SIOの書き込みレジスタ4の上位2ビット(D<sub>7</sub>, D<sub>6</sub>)でクロック・レートの設定を行いません。クロック・レートとは通信ボーレートと送信・受信クロック(TxC, RxC)の比率を示し

送信・受信クロック=ボーレート×クロック・レート

(1, 16, 32, 64)

という関係があります。ただし、クロック・レート1は外部同期にて使用します。

CTC, SIOの設定と通信ボーレートとの関係を表2に示します。

CTCの 分周比 (時間定数レジスタ値)	SIOのクロック・レートに対するボーレート(ボー)		
	×16	×32	×64
13	9600	4800	2400
26	4800	2400	1200
52	2400	1200	600
104	1200	600	300
208	600	300	150

表2 CTC、SIOの設定とボーレートとの関係

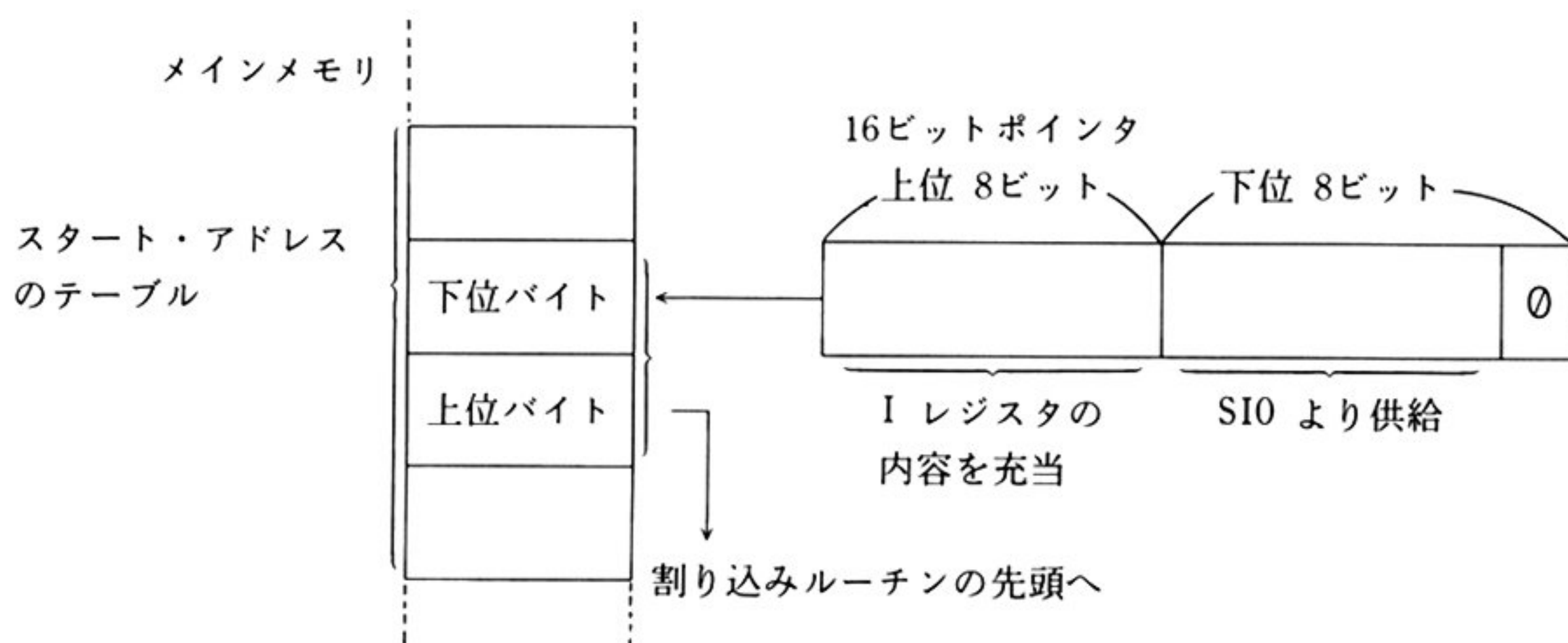
R S—2 3 2 C インターフェイスを使用してデータの送・受を行なう場合、一般的にはデータ信号線 (T x D, R x D)、コントロール信号線 (R T S, C T S, D T R, D S R) と G N D 線を使用します。これらの信号線は S I O のチャンネル A によりすべて制御できます。本機はこれらの信号線の他に C I (被呼表示)、C D (キャリア検出) を用意しています。この 2 つの信号線の状態は S I O のチャンネル B の入力ポートにより読み出せます。また、本機は内部同期だけでなく、外部同期にも対応できるように、S T 1 (送信信号エレメントタイミング)、S T 2 (送信信号エレメントタイミング)、R T (受信信号エレメントタイミング) も用意しています。内部/外部同期の切り換えは S I O のチャンネル B の D T R B 出力ポートとの H i g h / L o w により行ないます。S I O のチャンネル B はマウス用に使用しているため、D T R B 出力ポートを H i g h / L o w に切り換える時に R T S B 出力ポートの状態が変化しないように注意してください。

## 6

## 割り込み処理の使用

C P U と S I O の間で割り込み処理 (ベクトル割り込み) を使用する場合について説明します。

この割り込み処理を使用する場合、まえもってプログラムにより割り込みルーチンのスタート・アドレス (2 バイト) のテーブルをメモリの適当な位置 (偶数アドレスから下位バイト、上位バイトの順) に配置しておきます。C P U は割り込みを受け付けたとき 1 6 ビットのポインタで、必要な割り込みルーチンのスタート・アドレスをプログラムカウンタに読み込み、そのアドレスへジャンプします。このポインタ (割り込みスタート・アドレスのある位置を示している) として上位 8 ビットには I レジスタの内容を充当し、下位 8 ビットは割り込みアクノリッジ期間に S I O より割り込みベクトル (プログラムにより設定されていること) が自動的に供給されます。



本機の B A S I C を起動すると、I レジスタには & H F 8 が設定され F 8 3 0 ~ F 8 3 B 番地を割り込みスタート・アドレスのテーブルとして使用できます。



# 14章

## マウスインターフェイスの使い方

本機ではマウスインターフェイスが内蔵されており、コネクタもコンピュータ本体の前面トビラ内と後面の2ヶ所に用意されています。

### 1

### マウスとは

本機のマウスのハードウェアは、機械式に属します。

マウス裏面（ボタンのついた面の反対面）の中央に、大きな金属のボールがあり、そのボールがマウスの移動により回転し、その回転が本体内のローラに伝わり回転させます。そしてローラは、ロータリーエンコーダに回転を伝えます。これにより、ロータリーエンコーダではマウスの運転量に応じた信号を発生します。

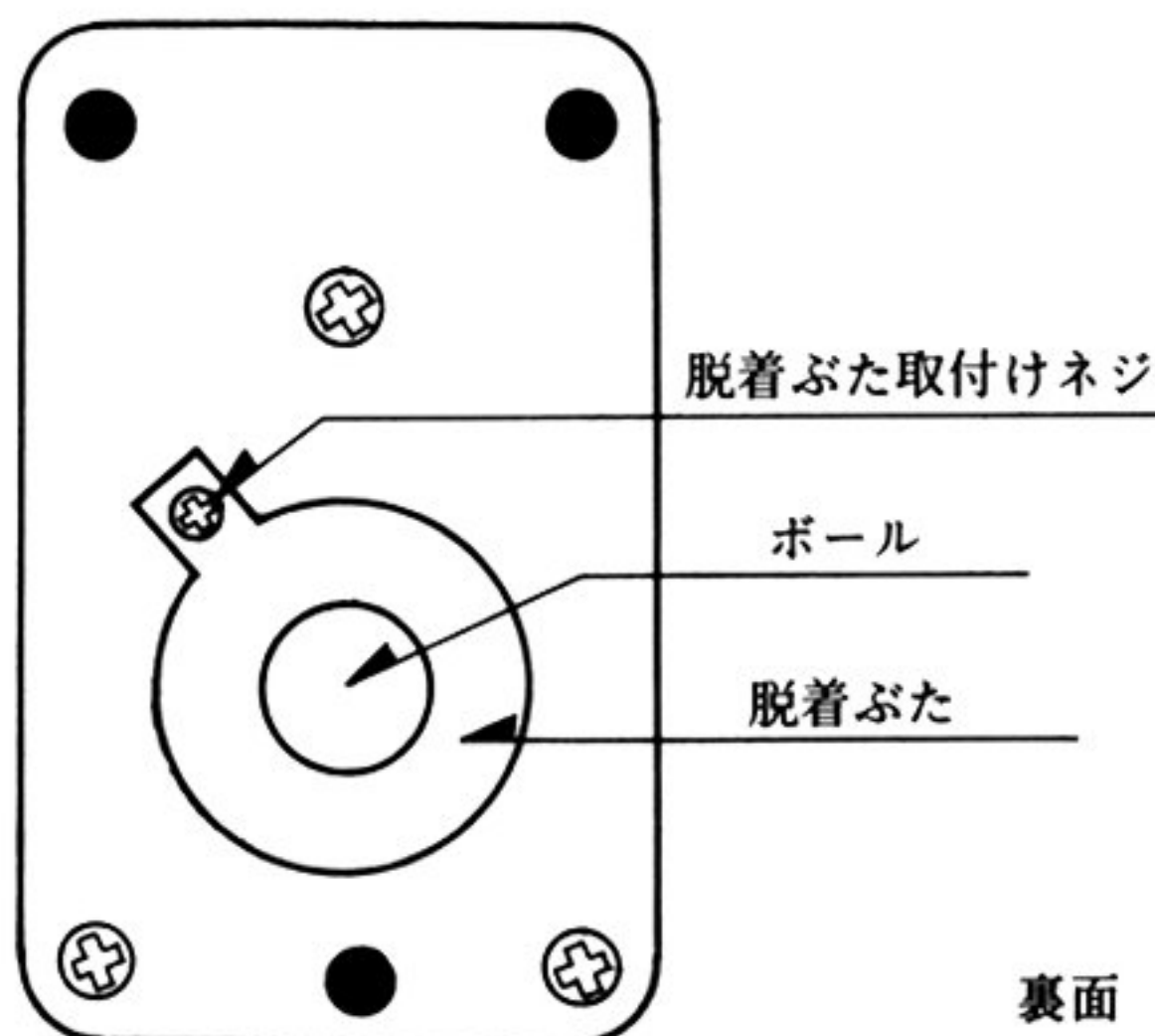
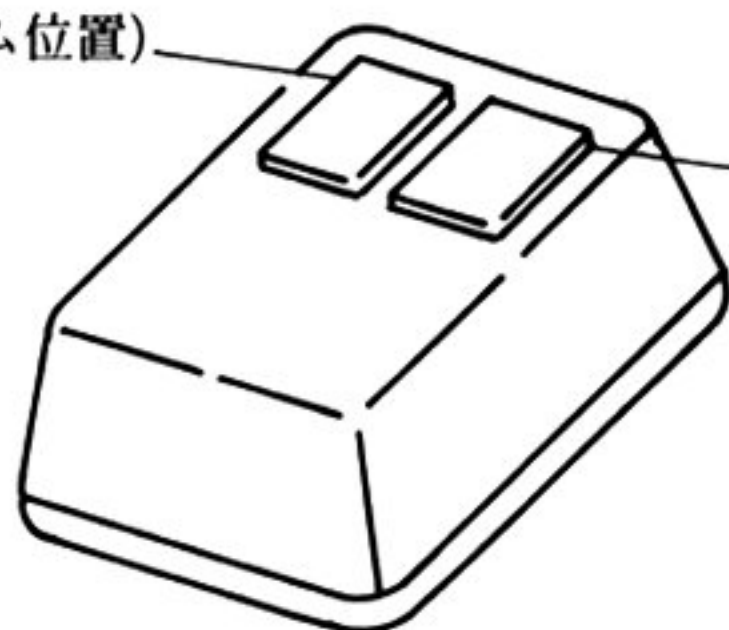
表面

1 側

(ホーム位置)

2 側

(座標入力)



マウスは、ローラとロータリーエンコーダがX軸（横）、Y軸（たて）方向用に1組ずつ用意されています。したがって、マウスの移動量が、X方向成分、Y方向成分に分けられ、X、Y座標の相対座標を表わして、そのため座標入力装置〈ポインティング デバイス〉とも言われます。

また、マウスには2つのボタンがついています。

このボタンは、機能の選択や指示などのセレクト用としてよく使われます。また、BASICでもマウス関数でサポートしています。ゲーム用として使う場合にはジョイスティックのトリガーボタンと同じ用途にも使えます。

### 2

### マウスを使う

本機では、ディスク BASICに、マウス関数が用意されています。以後この関数の使い方を中心に説明します。

# 14章

## 2.1 マウス関数の書式と機能について

### a. 機能の設定ステートメント

(1)

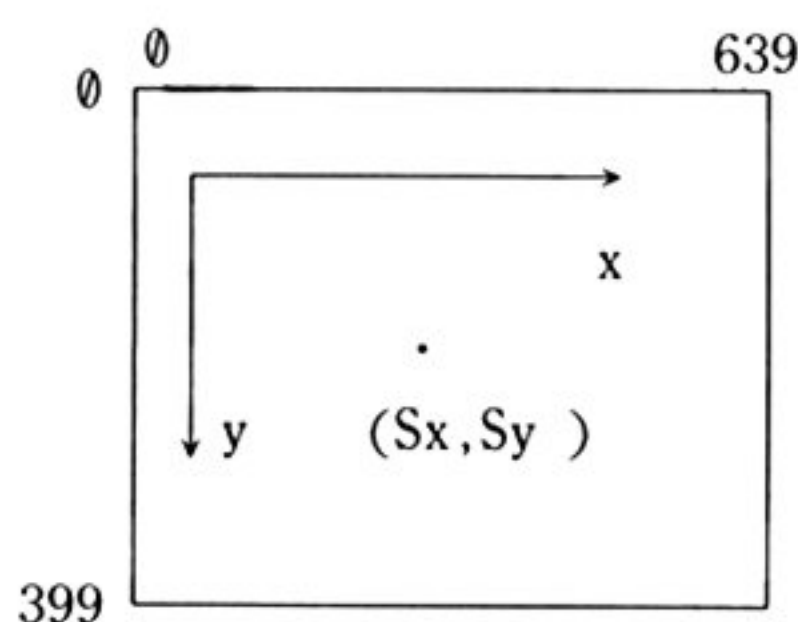
**MOUSE 0**

マウス機能をOFF状態にします。マウスを使用しないのにマウス機能をON状態にしておくと、割り込み<sup>注1)</sup>の関係で処理スピードが遅くなります。ですからマウスを使わない時は、このステートメントを実行してください。

注1) ここでいう割り込みとは、ある仕事をしていた途中で別の仕事をし、また元の仕事に戻る、といったことを指します。すなわちマウス機能がON状態だとすると、マウスを使う、使わないに関係なくマウスへの割り込みが発生するということです。別の実行をしているとき途中でマウスの割り込み処理が入るために実行が止まり、その分時間がかかってしまうのです。

(2)

**MOUSE 1, Sx, Sy**



マウス機能をON状態にし、初期座標として、画面座標 (Sx, Sy) を設定します。Sx、Syの値は、-32768~65535までの値が設定できます。しかし、グラフィック画面上への表示などを考えた場合、WIDTHステートメントによって設定した解像度と一致するように、設定するとよいでしょう。ただし、WIDTHステートメントの設定はOPTION SCREENステートメントによってエラーとなりますので、くわしくは『BASICリファレンスマニュアル』を参照してください。

Sxで設定された値がMOUSE関数のMOUSE (0) に、Syで設定された値がMOUSE (1) の初期値としてセットされます。

Sx、Syを省略するとマウス機能をON状態にするだけとなります。ただし、Sx、Syを省略した場合、BASIC起動時では、MOUSE (0)、MOUSE (1) には、それぞれ0が入ります。

また、Sx、Syを設定した場合、マウスマウスの移動範囲の設定 ((4) MOUSE 3の項参照) で、その値を超えていた時、移動範囲の最大値に、小さければ最小値に変わります。

(3)

**MOUSE 2, d, r**

X (横) 方向、Y (たて) 方向のマウスマウスの移動比率を設定します。dは、マウスマウスの移動方向を指し、d=0では、X (横) 方向、d=1では、Y (たて) 方向の移動比率の設定となります。移動比率rは、1~32の値を設定できます。rの値が大きくなるほど、マウスの移動に対する座標の変化は小さくなります。それは、マウスの移動量 (マウスから送られてくる値) を移動比率で割っているためです。つまりrの値が大きくなるほどMOUSE (0)、MOUSE (1)、に加えられる値が小さくなり、座標の変化が小さくなります。

BASIC起動時では、X、Y方向の移動比率rはそれぞれ10の値です。



(4)

**MOUSE 3, S<sub>x1</sub>, S<sub>y1</sub>, S<sub>x2</sub>, S<sub>y2</sub>**

マウスカーソルの移動範囲を設定します。

座標 (S<sub>x1</sub>, S<sub>y1</sub>) と (S<sub>x2</sub>, S<sub>y2</sub>) を結ぶ線を対角線とする四角形で囲まれた領域をカーソル移動範囲とします。グラフィック画面との関係から W I D T H ステートメントによって設定した解像度と一致させるとよいでしょう。

『BASICリファレンスマニュアル』の W I D T H、O P T I O N S C R E E N を参照してください。

また、(2)の項の M O U S E 1, S<sub>x</sub>, S<sub>y</sub> で設定された座標 (S<sub>x</sub>, S<sub>y</sub>) が、この移動範囲からずれていた場合、移動範囲にいちばん近い座標となります。実際は、移動範囲を出ていた方の値が移動範囲の最大値または最小値となります。

座標 (S<sub>x1</sub>, S<sub>y1</sub>) と (S<sub>x2</sub>, S<sub>y2</sub>) を設定するときは、S<sub>x1</sub> < S<sub>x2</sub>, S<sub>y1</sub> < S<sub>y2</sub> でなくてはなりません。

#### b. 状態の読み出し関数

(1)

**MOUSE (0)**

マウスカーソルの現在の X (横) 座標をこの値として返します。

(2)

**MOUSE (1)**

マウスカーソルの現在の Y (たて) 座標をこの関数の値として返します。

(3)

**MOUSE (2, b)**

指定されたボタン番号 b のボタンが押されているときに、-1 の値をこの関数の値として返し、押されていないときは、0 の値を返します。

ボタン番号 b とは、マウスのケーブルの出ている方向を上にして、

b = 1 で左側のボタンが押されている時のみ関数は -1

b = 2 で右側のボタンが押されている時のみ関数は -1

となり条件を満たさない場合この関数は 0 となります。

(4)

**MOUSE (3, b)**

ボタン番号 b のボタンが押された時のマウスカーソルの X (横) 座標をこの関数の値として返します。

(5)

**MOUSE (4, b)**

ボタン番号 b のボタンが押された時のマウスカーソルの Y (たて) 座標をこの関数の値として返します。

(6)

**MOUSE (5, b)**

ボタン番号 b のボタンが離された時のマウスカーソルの X (横) 座標を、この関数の値として返します。もちろんボタンは、押されていた必要があります。

(7)

**MOUSE (6, b)**

ボタン番号 b のボタンが離された時のマウスカーソルの Y (たて) 座標をこの関数の値として返します。

(8)

**MOUSE (7)**

最後にこの関数が呼び出されてから、次にこの関数が呼び出されるまでにマウスカーソルが動いた X (横) 方向の移動距離をこの関数の値として返します。つまり、最後にこの関数が呼び出された時の座標と、次のこの関数が呼び出された時の座標の X 座標の差がこの関数の値となります。

(9)

**MOUSE (8)**

最後にこの関数が呼び出されてから、次にこの関数が呼び出されるまでにマウスカーソルが動いた Y (たて) 方向の移動距離をこの関数の値として返します。つまり最後にこの関数が呼び出された時のマウスカーソルの座標と次にこの関数が呼び出された時のマウスカーソルの座標の Y 座標の差がこの関数の値となります。


## 2.2 マウス関数を動かす

---

次のプログラムを入力してみてください。

```
10 INIT:WIDTH 80,25,0,0
20 MOUSE1,0,0
30 MOUSE2,0,5
40 MOUSE2,1,12
50 MOUSE3,0,0,639,199
60 X=MOUSE(0):Y=MOUSE(1)
70 S1=MOUSE(2,1):S2=MOUSE(2,2)
80 PRINT "X= ";X,"Y= ";Y,"SW1= ";S1,"SW2= ";S2
90 GOTO 60
```



R U N  してみますと、下の様に表示されます。

X = 0 Y = 0 SW2 = 0


ここで、マウスを動かしてみてください。

X = と Y = の後の数値が変わります。

ではボタンを押したり離したりして見てください。

左側のボタンを押すと、「SW1 = -1」の表示が、右側のボタンを押すと、「SW2 = -1」の表示が現われ、離すとそれぞれ-1が0になることがわかります。

では、プログラムを止めてLIST  を入力してください。

プログラムの止め方は、 +  キーとしてください。

プログラムの説明をします。

行番号10は、画面を初期化し、画面座標系を640×200ドットの画面に設定します。専用ディスプレイテレビ以外のモニターをお使いの方は本機の標準／高解像動切換スイッチで合わせてください。プログラムを直す必要はありません。

行番号20は、マウス機能をON状態にし、初期座標を(0, 0)に設定しますので最初は、X = 0、Y = 0を表示します。

行番号30は、X (横) 方向の移動比率を5と設定します。

行番号40は、Y (たて) 方向の移動比率を12と設定します。

行番号30と40の移動比率(1～32の値)をいろいろ変えてみてください。違いがわかります。

行番号50は、マウスカーソルの移動範囲を(0, 0) - (639, 199)に設定します。プログラムを実行させた時、マウスを動かしてみてください。移動の範囲の値しかとれません。大きくなったり、小さくなったりはしません。

行番号60は、変数XにマウスカーソルのX(横)座標の値を、変数YにマウスカーソルのY(たて)座標の値を代入します。

行番号70は、変数S1に左側のボタンの状態を変数S2に右側のボタンの状態を代入します。どちらもボタンが押されていると-1の値が、離されているとき0の値が入ります。

行番号80は、それぞれの変数と変数の値の表示をします。

行番号90は、60にジャンプします。

## 2.3 マウスカーソルを表示させる

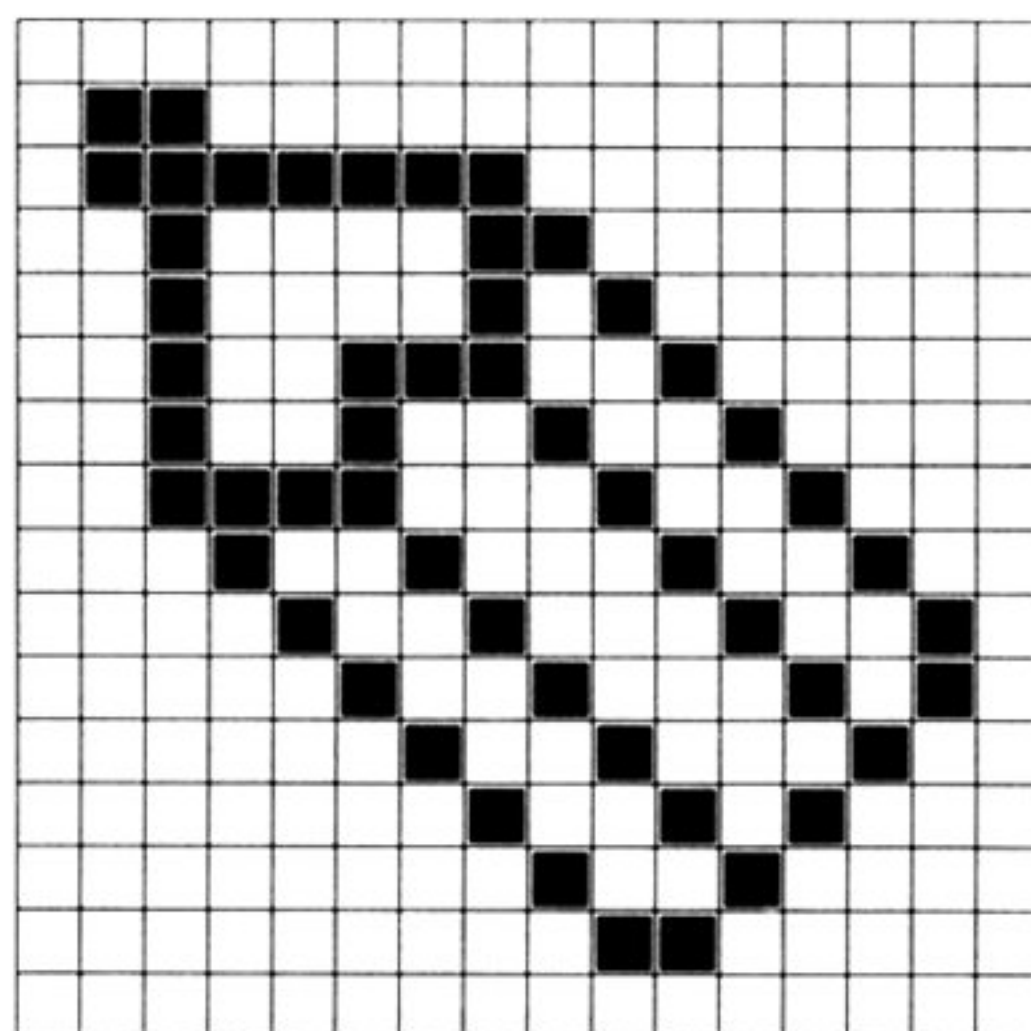
---

本機ではマウスカーソルを表示するには、そのためのプログラムの入力が必要です。次に示すのがそのサンプルです。図1の様なエンピツ状のマウスカーソルを表示し、マウスの左のボタンを押しながらマウスを動かすと線を引き右のボタンで画面をクリアするというものです。

```

10 INIT:PALET:OPTIONSCREEN0
20 WIDTH 80,25,1,2
30 LINE(0,0)-(7,7),XOR,BF,HEXCHR$("000000606060 7F7F7F 212121 212121 272727 242424 3C3C3C")
40 LINE(8,0)-(15,7),XOR,BF,HEXCHR$("000000 000000 000000 808080 404040 202020 909090 484848")
50 LINE(0,8)-(7,15),XOR,BF,HEXCHR$("121212 090909 040404 020202 010101 000000 000000 000000")
60 LINE(8,8)-(15,15),XOR,BF,HEXCHR$("242424 121212 8A8A8A 444444 282828 909090 606060 000000")
70 DIM A(50)
80 GET@(0,0)-(15,15),A,7
90 X=0:Y=0:XX=0:YY=0
100 MOUSE1,0,0
110 MOUSE2,0,5
120 MOUSE2,1,4
130 MOUSE3,0,0,624,384
140 PUT@(X,Y)-(X+15,Y+15),A,XOR,7
150 X=MOUSE(0):Y=MOUSE(1)
160 IF MOUSE(2,1)=-1 THEN LINE(XX,YY)-(X,Y),PSET
170 PUT@(X,Y)-(X+15,Y+15),A,XOR,7
180 XX=X:YY=Y
190 IF MOUSE(2,2)=-1 THEN CLS0:GOTO 150
200 GOTO 140

```



データ

00	00
60	00
7F	00
21	80
21	40
27	20
24	90
3C	48
12	24
09	12
04	8A
02	44
01	28
00	90
00	60
00	00

図1 マウスカーソルの形状例とデータ (16進数)

プログラムの説明をします。

行番号10は、画面の初期化です。

行番号20は、グラフィック画面の解像度を640×400ドットにします。ですから専用ディスプレイテレビまたは高解像度ディスプレイをお使いでない方はWIDTH 80, 25, 0, 0に直してください。ただし、マウスカーソルの形状は画面のたて、横比の関係で少しゆがみます。30～60は、マウスカーソルの形状をグラフィック画面の座標(0, 0)と(15, 15)を対角点とする四角形の中に表示します。実際は見えません。

行番号70は、配列Aを51個用意します。

行番号80は、先ほどグラフィック画面に表示したマウスカーソルの形状を配列Aに読み込みます。

行番号90は、変数の初期値を0とします。

行番号100は、マウス機能をON状態にし、初期座標を(0, 0)に設定します。

行番号110は、X(横)方向の移動比率を5と設定します。

行番号120は、Y(たて)方向の移動比率を4と設定します。



行番号130は、マウスカーソルの移動範囲(0, 0) - (624, 384)に設定します。ここで、注意して頂きたいのは、グラフィック画面の座標(0, 0) - (639, 399)と一致していない点です。これは、PUT@ステートメントの有効範囲がこの場合、X方向639、Y方向399となっており、行番号140のステートメントがX+15, Y+15になっています。マウスの座標X、Yが(625, 385)を超えると、PUT@の有効範囲を超えてエラーとなります。行番号20でWIDTH 80, 25, 0, 0とした方は、MOUSE 3, 0, 0, 624, 184としてください。

行番号140、170は、80で読み込んだ図形を表示します。

行番号150は、XにマウスのX座標をYにマウスのY座標を代入します。

行番号160は、マウスの左のボタンが押されていると線をひきます。

行番号180は、XXにXの値をYYにYの値を代入します。

行番号190は、マウスの右のボタンが押されると画面をクリアします。

行番号200は、140にジャンプします。

### 3

## 使用上の注意事項

本機にマウスを接続する場合、コンピュータ本体の前面トビラ内と後面の2箇所のコネクタに同時に取り付けて実行することはおやめください。正常な動作をしません。

### 参 考

グラフィックツール・嬉楽画ターボ・CZ-114SFにはマウスが付属されています。内容については「15章デジタルテロPPERの使い方」の

**参考**をお読みください。

# 15章

## デジタルテロッパーの使い方

本機では、専用ディスプレイテレビとの組み合わせ、あるいは家庭用テレビとの組み合わせにより、コンピュータ画像とテレビやビデオ画像との重ね合わせ〈スーパーインポーズ〉画像が表示できます。さらに、内蔵のデジタルテロッパー機能を利用すれば、コンピュータ画像やスーパーインポーズ画像のビデオ画像の録画が可能になります。

### 1

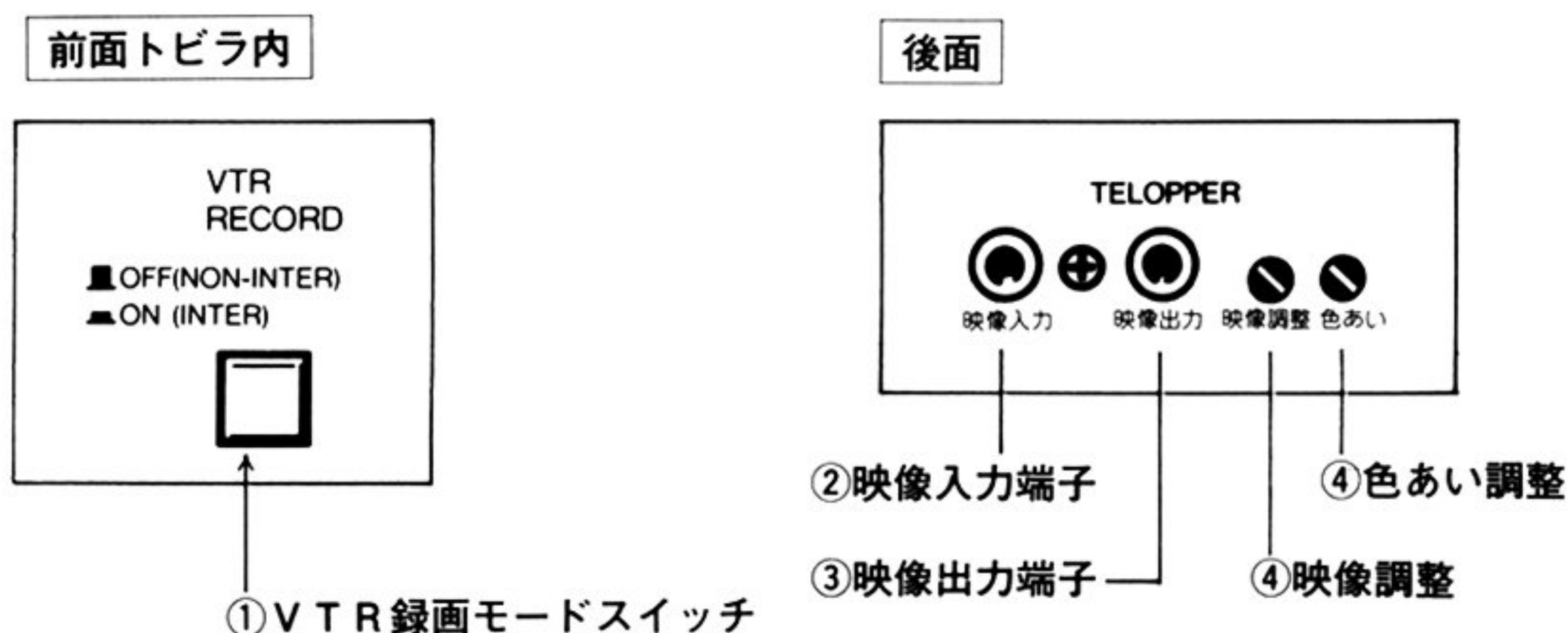
### 特長

スーパーインポーズ処理部にSSS（セパレート・サブキャリア・スーパーインポーズ）方式を採用しました。

- コンピュータ映像部のエッチノイズ（輝度、色、フリッカー）が極めて少なく安定高画質なスーパーインポーズを実現しました。
- 入力映像ソースの画像劣化がほとんどありません。
- VTR等不安定な映像ソースにも、安定した同期で、かつ、コンピュータ映像レベル、色あいを保持するダブルクランプ方式（ピーク&ペデスタルクランプ）を採用しました。

### 2

### 各部名称



#### ① VTR録画モードスイッチ

コンピュータ画像をVTRに録画するときはスイッチをON（インターレースモード）にします。詳しくは「**3** VTR録画モードスイッチについて」を参照してください。

#### ②映像入力端子

接続機器（テレビ、VTR、ビデオディスク、ビデオカメラなど）からの映像信号を入力する端子です。



### ③映像出力端子

コンピュータ画像（NTSC信号に変換されたもの）、スーパーインポーズ画像、ビデオ画像の出力を行なう端子です。

### ④映像調整、色あい調整

映像出力端子より出力されるコンピュータ画像の調整ボリュームですが、あらかじめ、標準状態に設定してあります。調整したい場合は販売店にご相談ください。

## 3

## VTR録画モードスイッチ

本機前面のトビラ内にはVTR録画モードスイッチがありますが、これには2つのモードがあります。以下このモードについて説明します。

このスイッチは、コンピュータ画面にしたとき使用し、その画像を録画する場合にはONにし、録画しない場合はOFFとします。

なお、テレビ（またはビデオ）画面、スーパーインポーズ画面にした場合は、自動的にインターレースモードになります。

	VTR録画モードスイッチOFF (■) (ノンインターレースモード)	VTR録画モードスイッチON (■) (インターレースモード)
使用目的	• コンピュータ画像をVTRに録画しないとき（プログラム作成時など）はOFFにしておきます。	• コンピュータ画像をVTRに録画するときはONにします。
出力信号 形 態	• コンピュータ本体内部で発生する同期信号を使って出力するので、NTSC標準複合信号から少しはずれた信号です。（内部同期） • ノンインターレース走査	• コンピュータ本体へ入力する映像信号の同期信号を使って出力するので正規のNTSC標準複合信号です（外部同期） • インターレース走査
映像入力	不 要	必 要
特 長	• ノンインターレース走査のため映像の垂直ゆれ（フリッカー）がありません。	• 正規のNTSC標準複合信号ですからVTRに録画が可能です。 • 白文字に着色現象がありません。 • モード切り換え（スーパーインポーズ／コンピュータ）を行なっても同期の乱れがありません。
注意事項	• 白文字に着色現象があります。 • くし形フィルター方式のテレビと本機とを接続した場合には、一般のテレビよりもコンピュータ画面に、にじみがでます。	• VTRを映像入力として使用した場合、VTR再生時以外（停止、早送り、巻戻しなど）で画像に乱れが生じることがあります。

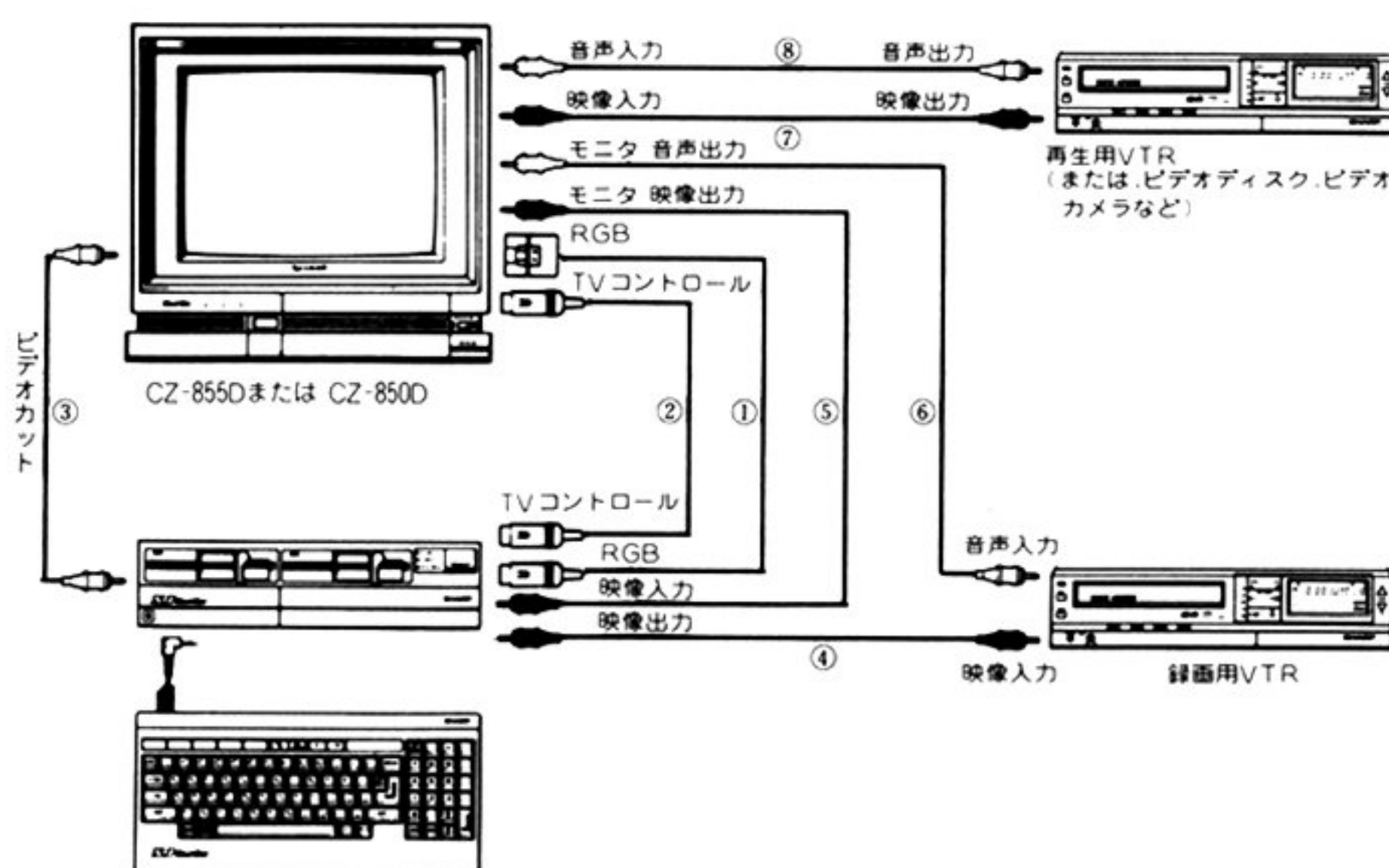
本機内蔵のデジタルテロッパーにより映像出力端子から次の3種類の信号が出力されます。

- ① N T S C 信号に変換されたコンピュータ画像の出力。(ノンインターレースモードとインターレースモードの2種類あります。)
- ② 映像入力端子に入力された N T S C 信号とコンピュータの R G B 信号を一つの N T S C 信号に合成・変換したスーパーインポーズ画像の出力。
- ③ 映像入力端子に入力された N T S C 信号をそのまま出力。

接続については、ご使用になる接続機器 (V T R、ビデオディスク、ビデオカメラ、ビデオ入出力端子付テレビなど) の組み合わせによってさまざまな接続方法がありますが、ここでは標準的な接続方法の2通りを説明をします。

接続する時には各接続機器の電源をかならず O F F (切) にしてください。なお接続用ケーブルは最寄りの販売店でお買い求めください。

#### (1) 専用ディスプレイテレビとの組み合わせ例



#### 接続方法

- ① 本機とディスプレイテレビとを R G B 信号用ケーブル、テレビコントロールケーブル、ビデオケーブルで接続します。
- ② 本機の映像出力端子と録画用 V T R の映像入力端子とを接続します。
- ③ 本機の映像入力端子とディスプレイテレビのモニター出力端子 (映像) とを接続します。
- ④ ディスプレイテレビのモニター出力端子 (音声) と録画用 V T R の音声入力端子とを接続します。
- ⑤ ディスプレイテレビの映像入力端子と再生用 V T R の映像出力端子とを接続します。
- ⑥ ディスプレイテレビの音声入力端子と再生用 V T R の音声出力端子とを接続します。



## 操作方法

接続ができましたら、各接続機器の電源をON（入）にしてください。なお、各接続機器の取り扱いや操作については、それぞれの取扱説明書にしたがってください。

- ①コンピュータ本体前面トビラ内のVTR録画モードスイッチをON（入）にしてください。
- ②テレビ放送を録画したいときは、ディスプレイテレビCZ-855DまたはCZ-850Dをテレビ画面にし、再生用VTRの映像を録画したいときは、ビデオ画面（画面表示はP.）にします。（コンピュータ画像のみを録画したい場合、テレビ画像またはビデオ画像などのNTSC信号をコンピュータの映像入力端子に入力する必要があります。）

### ●コンピュータ画像のモニター録画する場合

本機のキーボードで **SHIFT** キーを押しながらテンキーの **[・]** キーを押すとディスプレイテレビにはコンピュータ画像が映し出され、録画できる状態になります。

### ●スーパーインポーズ画像のモニター・録画をする場合

本機のキーボードで **SHIFT** キーを押しながらテンキーの **[+]** キーを押すとディスプレイテレビにはスーパーインポーズ画像が映し出され、録画できる状態になります。

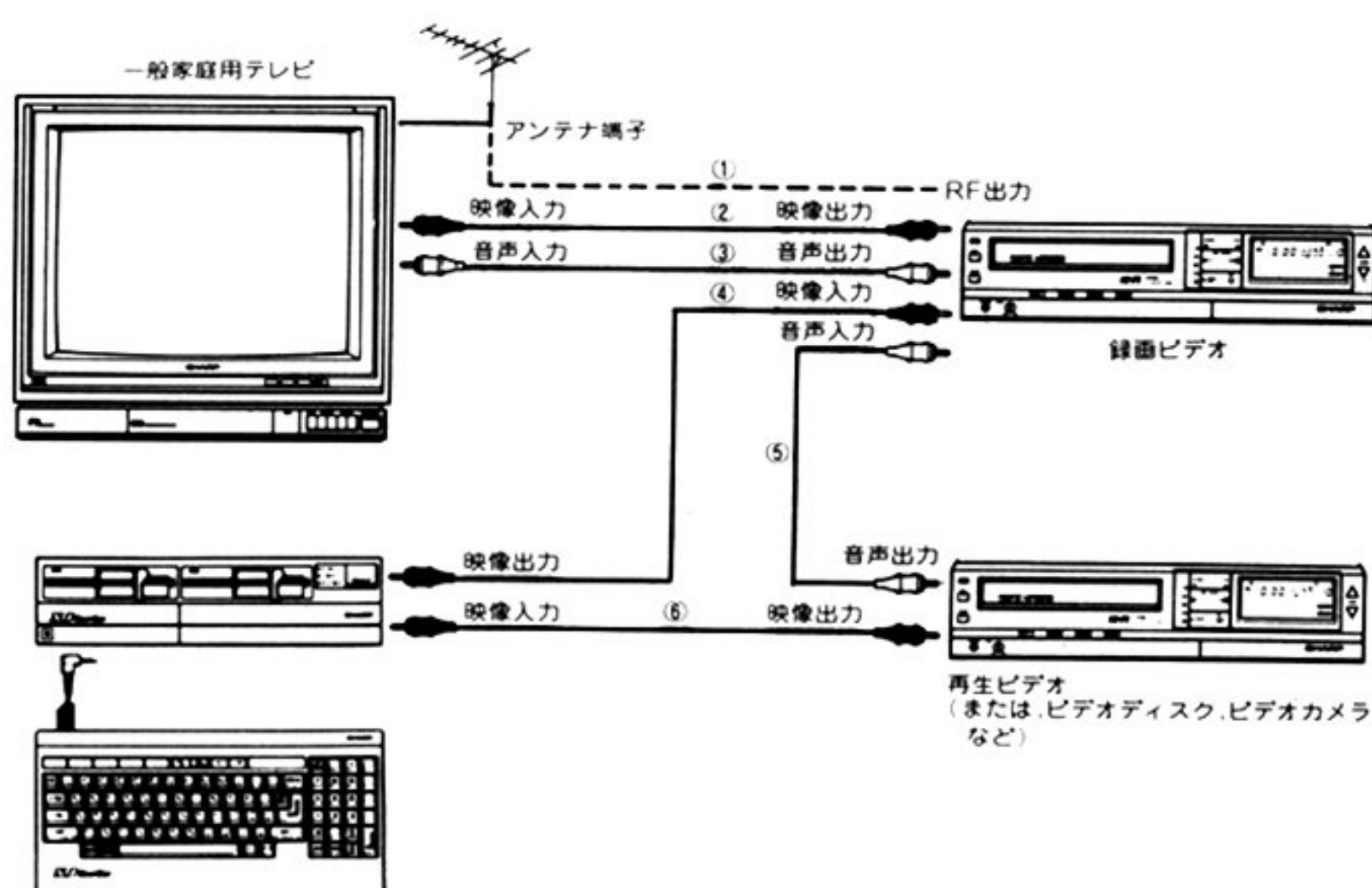
### ●ビデオ画像のモニター・録画をする場合

本機のキーボードで **SHIFT** キーを押しながらテンキーの **[=]** キーを押すとディスプレイテレビにはテレビまたはビデオ画像が映し出され、録画できる状態になります。

### ●プログラムを組む場合

前面トビラ内のVTR録画モードスイッチをOFF（切）にし、本機のキーボードで **SHIFT** キーを押しながらテンキーの **[・]** を押します。

## (2)一般家庭用テレビとの組み合わせ例



＊映像入力端子付テレビの場合は②、③の接続を行ない、端子がない場合には①の接続を行ないます。

### 接続方法

- ①テレビに映像入力端子がない場合、この接続を行ないます。録画用VTRのRF出力とテレビのアンテナ端子を接続します。
  - ②テレビに映像入力端子がある場合、この接続を行ないます。録画用VTRの映像出力端子とテレビの映像入力端子を接続します。
  - ③テレビに音声入力端子がある場合、この接続を行ないます。録画用VTRの音声出力端子とテレビの音声入力端子を接続します。
  - ④本機の映像出力端子と録画用VTRの映像入力端子を接続します。
  - ⑤再生用VTRの音声出力端子と録画用VTRの音声入力端子を接続します。
  - ⑥再生用VTRの映像出力端子と本機の映像入力端子を接続します。
- 以上の接続ができましたら、各接続機器の電源をON（入）にして次のように操作してください。  
なお、各接続機器の取り扱いについては、各々の取扱説明書にしたがってください。

### 操作方法

専用ディスプレイテレビとの組み合わせ例の操作方法を参照してください。

ただし、②項については、次に従ってください。

- ② 接続の手順で、①を行なった場合は、チャンネルを1chまたは2chに合わせます。
- ②③を行なった場合は、ビデオ画面にします。

## 5

### 黒抜き表示

本機では、テキスト画面、グラフィック画面ともに黒抜き表示が可能です。つまり、テキスト画面ではグラフィック画面、テレビ画面に対する黒抜き表示が行なえ、グラフィック画面ではテキスト画面、テレビ画面に対する黒抜き表示が行なえます。

黒抜き表示は、専用ディスプレイテレビ（CZ-855DまたはCZ-850D）との組み合わせでは双方のビデオカット端子を接続することによってRGB出力で直接行なうことができます。それ以外のモニターでは本機の映像出力端子から出力されるNTSC信号によって、黒抜き表示を行うことができます。

## 6

### 注意事項

- ①このテロップ機能は標準ディスプレイモードのときのみ使用可能で高解像度ディスプレイモードでは使用できません。
- ②コンピュータ画像またはスーパーインポーズ画像をVTRに録画する際はNTSC信号という帯域制限を受けるため、コンピュータ画面を下のいずれかに設定してください。

40字×10行、40字×12行、40字×20行、40字×25行

また、タイリングペイントの様な細いドット構成の画面はちらつき、色変化を生ずることがあります。さらにVTRの性能により録画画像に色ずれ、画質の劣化を生ずることがあります。

（理由）

RGB信号の周波数特性は10—14MHzであるのに対し、NTSC信号は下図に示すように輝度信号の後ろにある色信号と交錯しないようにするため、3—3.2MHz程度の帯域巾となっています。このため横80字（水平640ドット）のように細い指定をしても見にくかったり、色つきが悪かったりします。



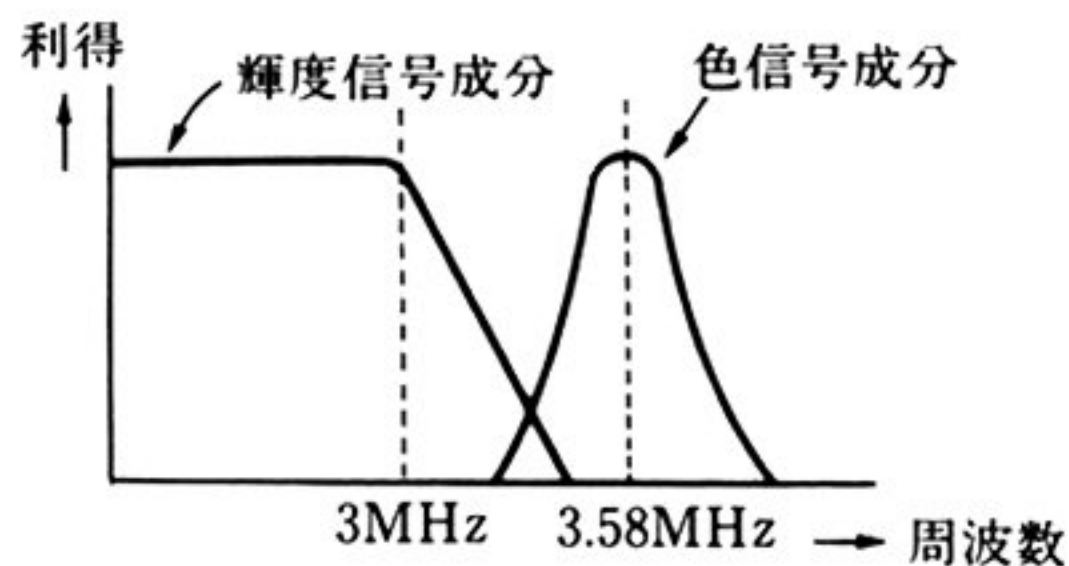
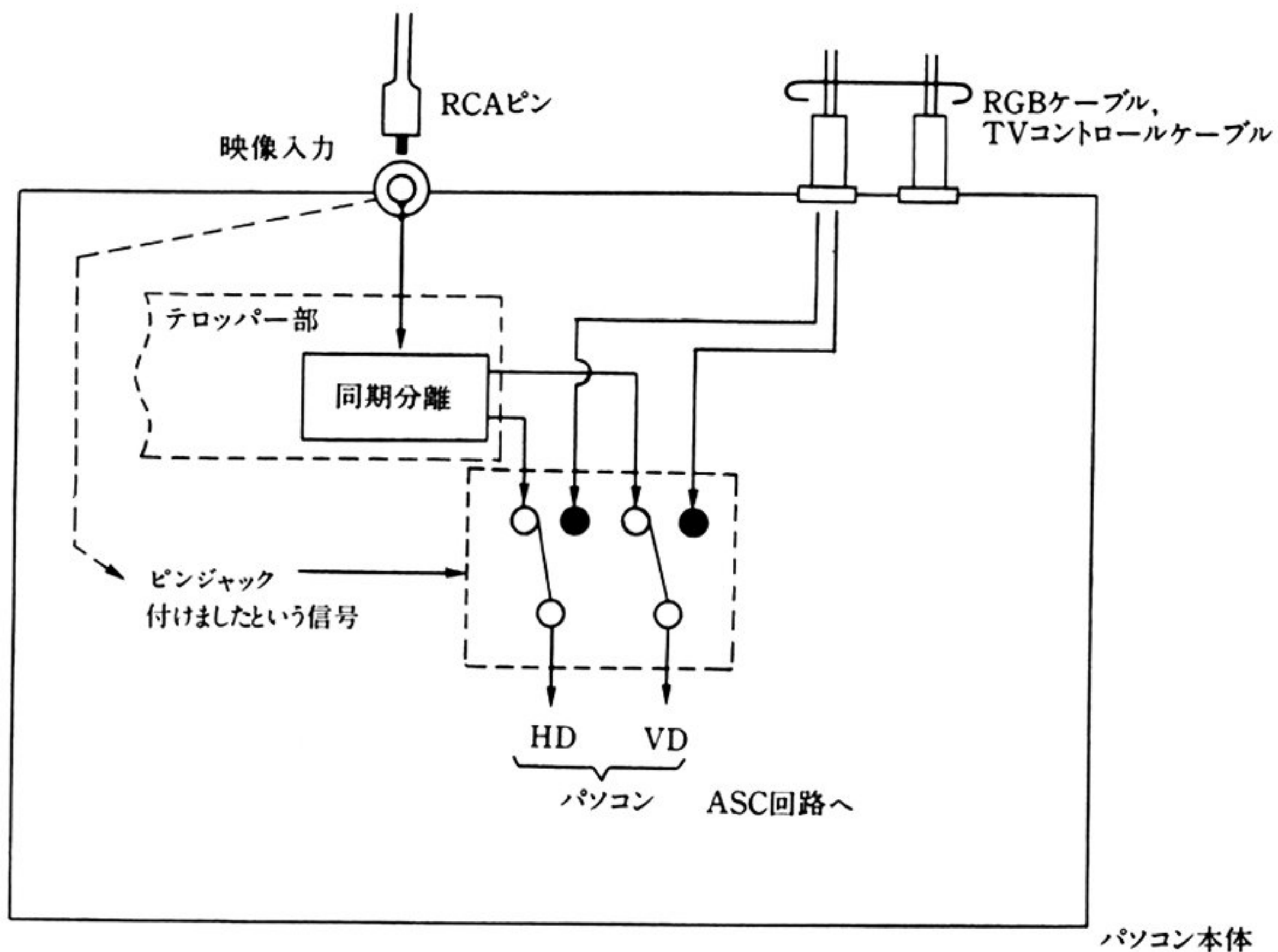


図. NTSC信号の周波数スペクトラム

- ③本機の映像入力端子にRCAピンケーブルを接続し、そのケーブルが他の映像機器（VTR、ビデオカメラ、ビデオディスク等）の映像出力端子に接続されていない場合や、接続されていても、映像信号が本機に入力されている場合、本機の映像出力信号はもとより、モニターの画像も乱れます。従ってデジタルテロッパー使用の際は、必ず映像入力端子に他の機器より映像信号を入力してください。もし、デジタルテロッパーを用いない場合にはVTR録画モードスイッチをOFF（切）にするか、もしくは、本機の映像入力端子からRCAピンケーブルをはずしてください。



(理由)

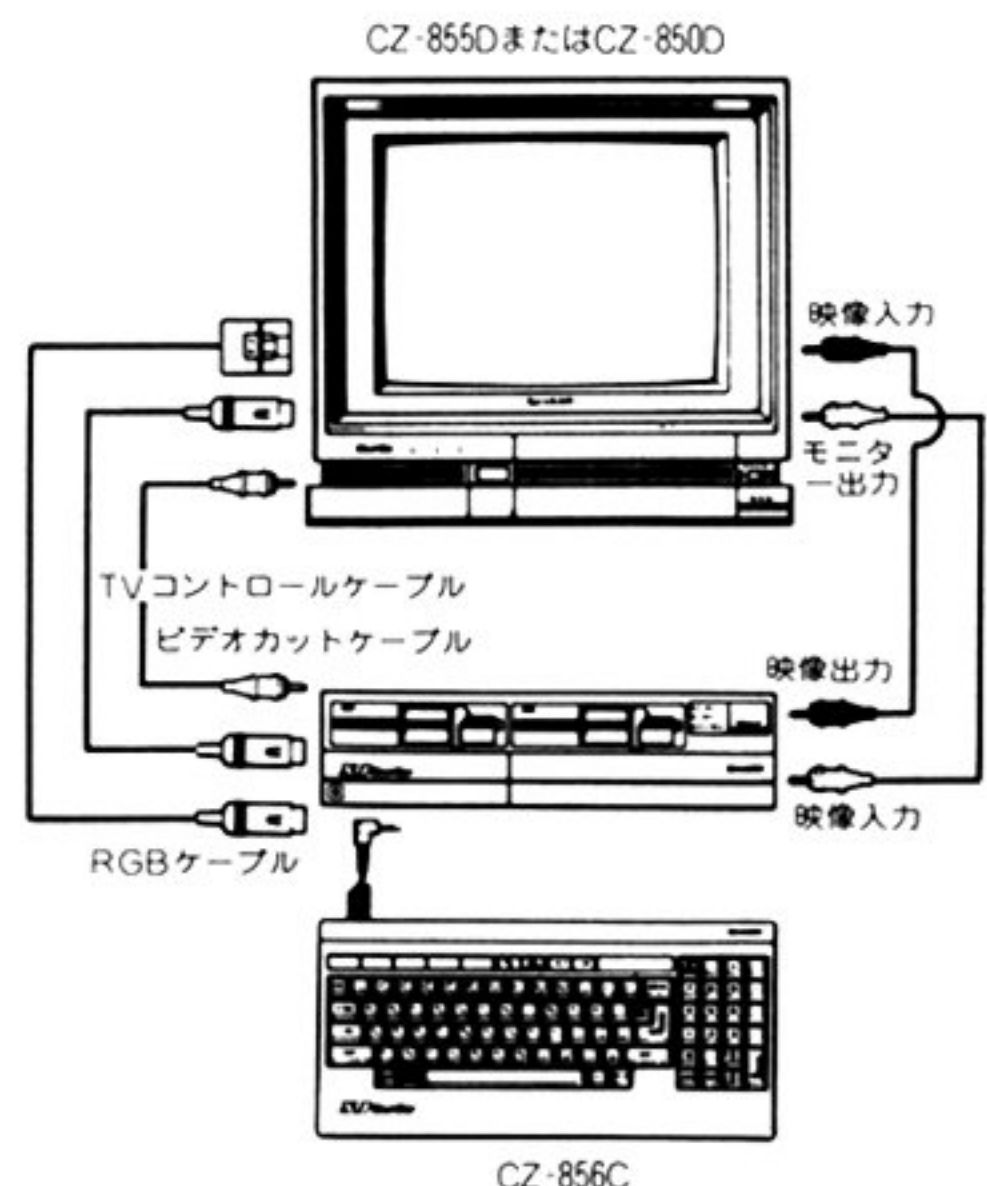
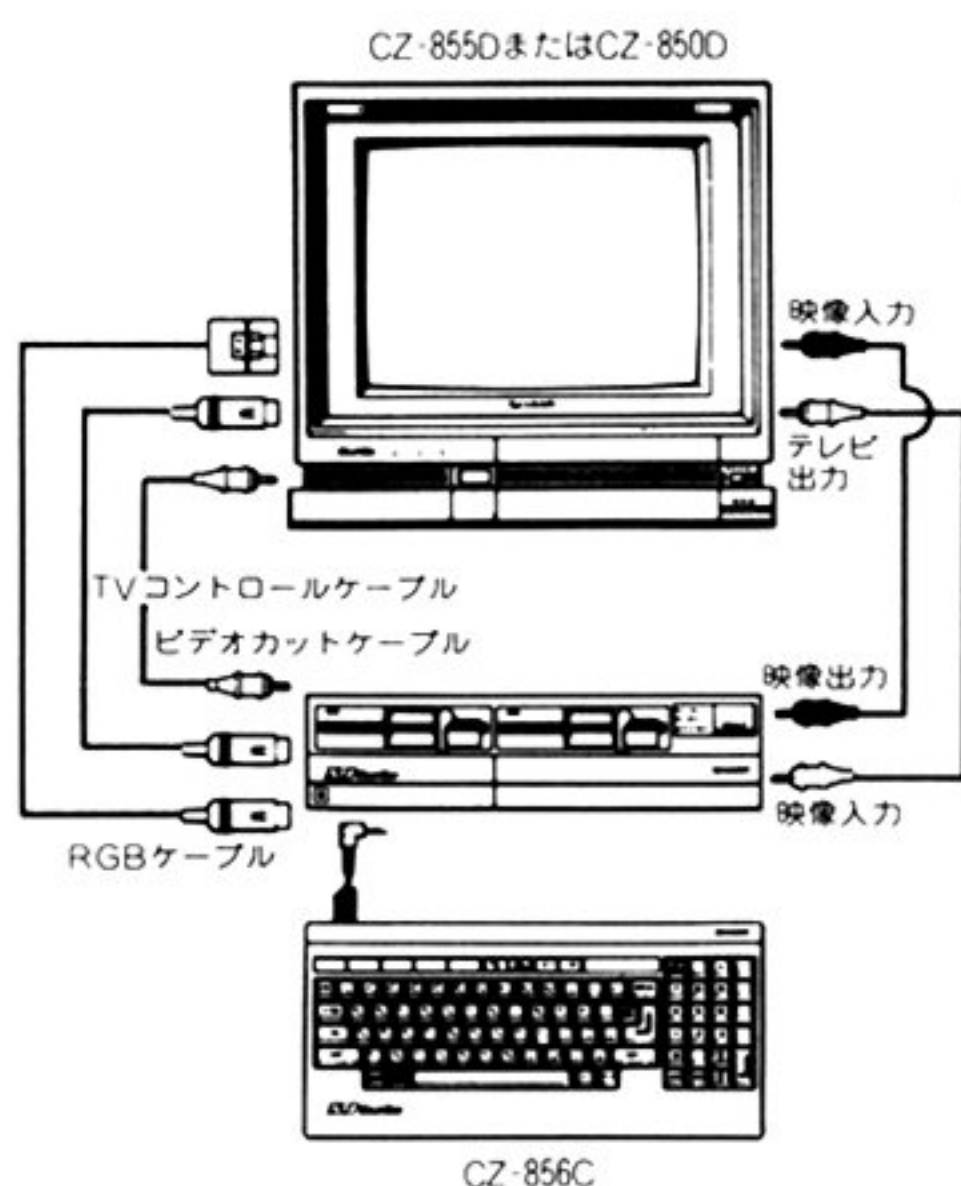
スーパーインポーズは映像信号にRGB信号を同期させて行なっています。専用ディスプレイテレビと本機との組み合わせでは、ディスプレイテレビの水平同期（HD）信号、垂直同期（VD）信号にRGB信号を同期させてスーパーインポーズを行ないます。一方、外部より映像機器（VTR、ビデオディスク、ビデオカメラなど）を接続し、ビデオ画面とのスーパーインポーズを行なう場合、本機の映像入力端子と映像機器の映像出力端子とをRCAピンケーブルで接続しますが、このとき本機内部では、RCAピンジャックを差し込んだ時から、映像機器からHD、VD

信号を供給することになります。したがって、本機の映像入力端子にRCAピンジャックを差し込んで、映像機器に接続されないまま放置されると、HD、VD信号が本機に入らなくなり、画面がみだれて何も見えなくなります。

- ④映像入力端子にモニターの映像信号と異なる、映像信号が入力されている場合スーパーインポーズモードにしますとモニター画面が乱れます。モニターのスーパーインポーズ画面を使用するときは映像入力端子にはモニターと同一の映像信号を入れるかまたは何も接続しないでください。
- ⑤本機の映像入力端子へ入力される映像が白黒であれば、スーパーインポーズ時またはコンピューターモード時（インターレースモード）のときのコンピュータ画像は色がつかなかったり、部分的に着色したりすることがあります。ただし映像が白黒であってもカラーバースト信号がついていればカラーになります。
- ⑥本機に入力される映像信号の同期部分が劣化している場合、コンピュータ画像が乱れることがあります。たとえば、テレビのアンテナ入力信号が弱い場合のテレビ出力や幾度もダビングされたVTRテープの再生信号等。
- ⑦次記のような接続では、画像が異常になるため使用しないでください。二重スーパーインポーズとなります。

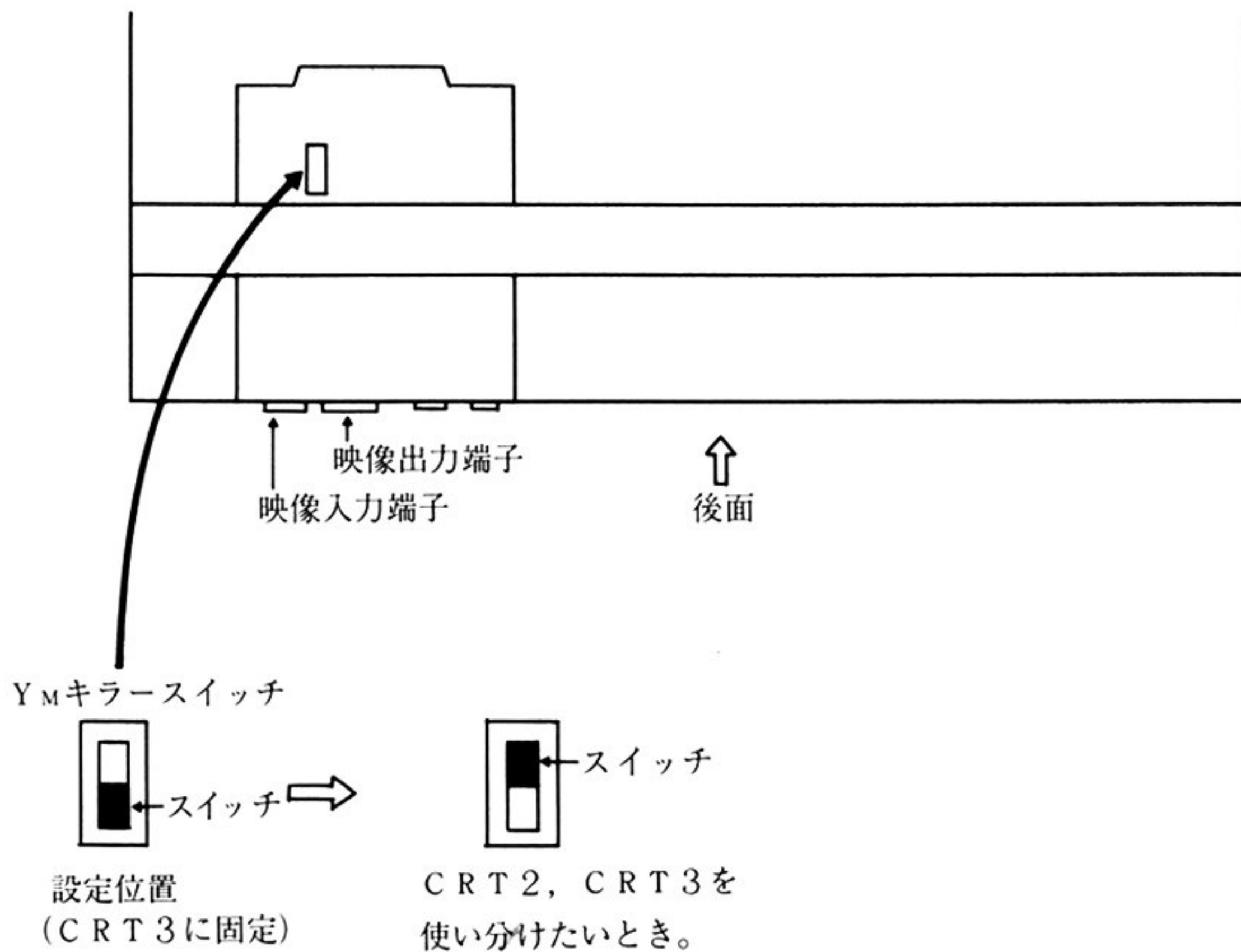
(i)二重スーパーインポーズとなります。

(ii)専用ディスプレイテレビ（CZ-855DまたはCZ-850D）をビデオモードにしたとき、同期流れがおこります。



- ⑧本機の映像出力端子より出力されるスーパーインポーズ画像はBASICステートメントCRT 3（テレビ放送とコンピュータ画面を同時に重ねて表示）を実行した状態になっています。プログラムにて映像出力をCRT 2（テレビ放送のコントラストを下げて、コンピュータ画面を重ねて表示）状態とCRT 3の状態とを使い分けたい場合は、上ぶたを開けて、YMキラースイッチをコンピュータ前面側へスライドさせてください。

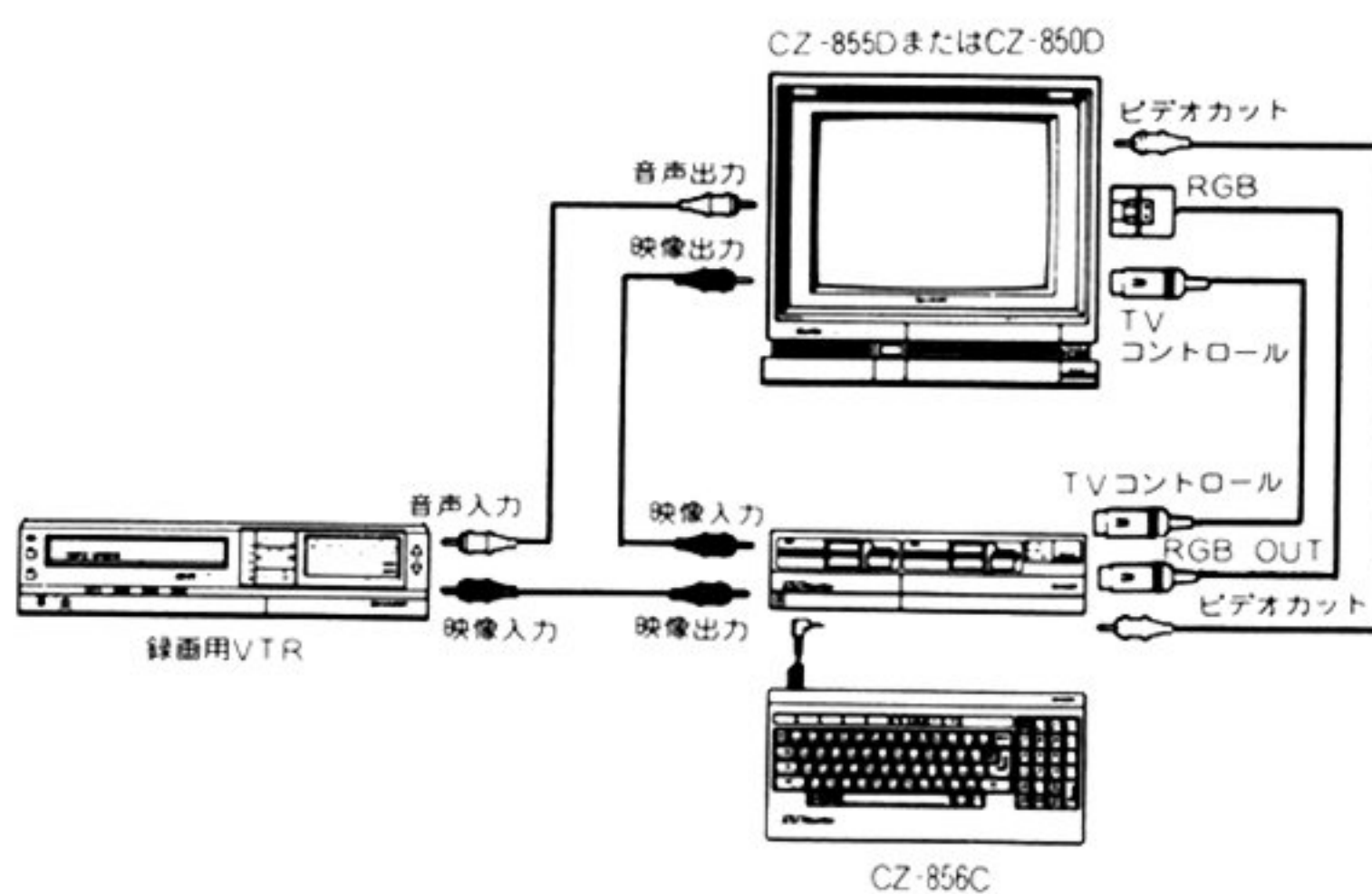




# 参考

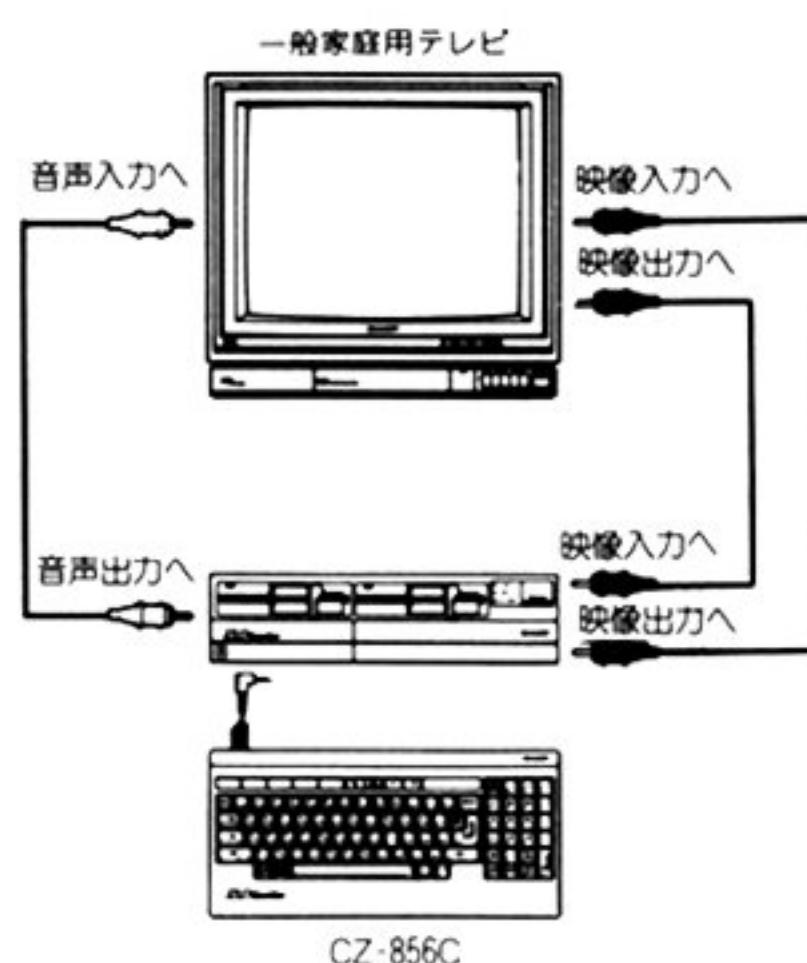
## 〈接続応用例①〉

- 専用ディスプレイテレビ (CZ-855DまたはCZ850D) を使用して "コンピュータ画像" "テレビとのスーパーインポーズ画像" "テレビ画像" をVTRに録画する場合。



### 〈接続応用例②〉

- 家庭用テレビ（ビデオ入出力端子付）に接続し、テレビ画像とコンピュータ画像を重ね合わせ、スーパーインポーズ画像を楽しむ場合。



### 〈ビデオマルチプロセッサ CZ-8VP1との接続例〉

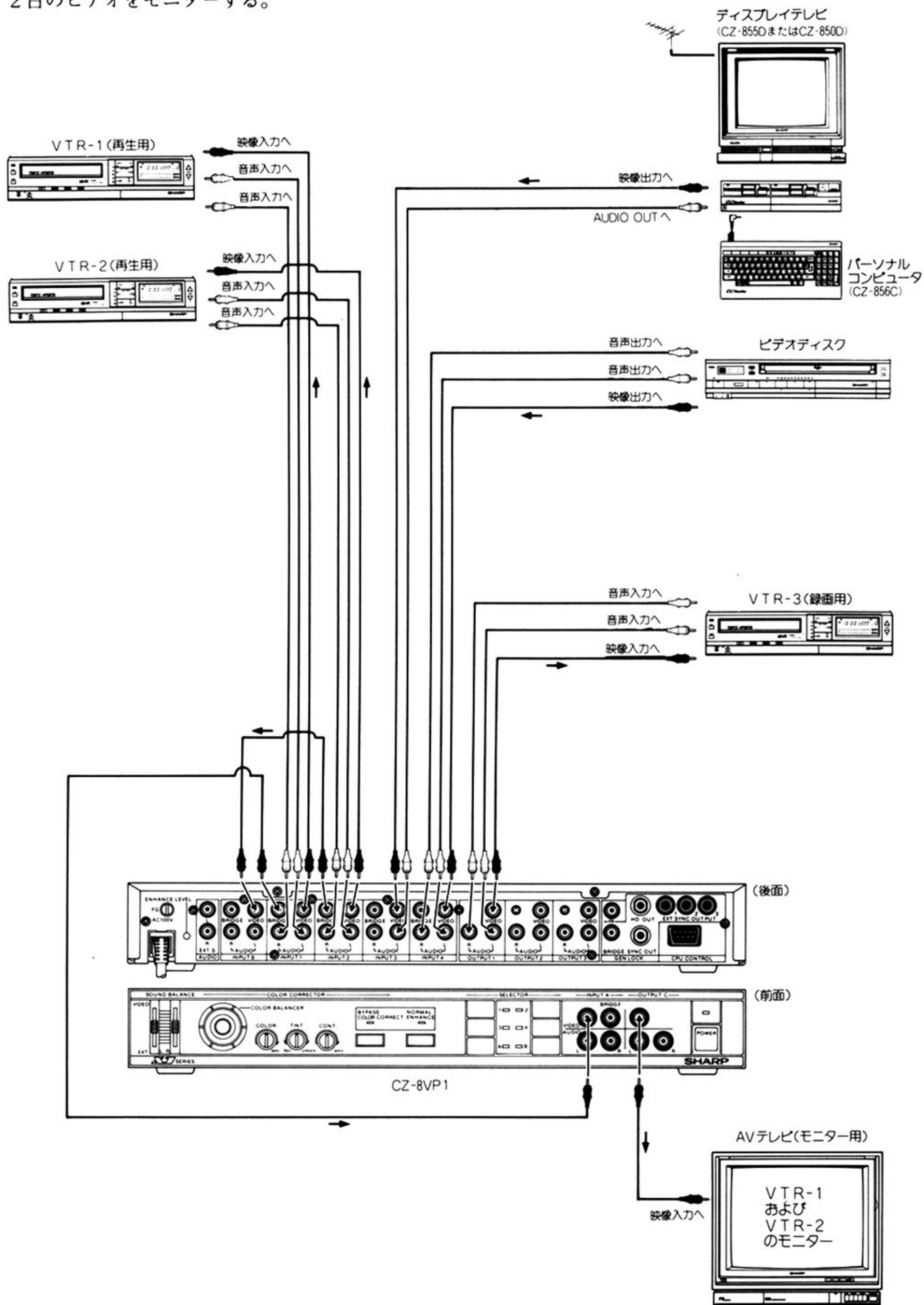
オプションのビデオマルチプロセッサ CZ-8VP1は、A/Vスイッチャー、カラーコレクタ、ビデオエンハンサ、GENLOCK機能等を装備し、本機との組み合わせにより、コンピュータコントロールが可能な映像機器です。特長は次のとおりです。

- ① 4入力3出力系と2入力1出力系のA/Vスイッチャーを独立に2基搭載しています。また、6系統の入力にはすべてブリッジ（共同接続）端子を設けました。
- ② オーディオ系は、すべてHi-Fiステレオ対応。また、左右独立のサウンドバランスつまみを装備しました。
- ③ 映像のカラー信号を安定再生するカラーコレクター機能を装備しました。
- ④ ディレイライン方式のビデオエンハンサ機能を装備しました。
- ⑤ ABロール編集（複数のビデオを再生して編集）ができるゲンロック端子を装備しました。
- ⑥ 入力信号切換やカラーコレクタ、エンハンサのON/OFFがコンピュータでコントロールできます。



# 〈接続例〉

パソコン、ビデオ、ビデオディスク等、4つのソースを切り換えて録画し、またこれとは独立して2台のビデオをモニターする。



### 〈グラフィックツール・嬉楽画ターボ・CZ-114SF〉

マウスを使って、コンピュータグラフィックスやタイトル文字の作成などを行うソフトウェアで、作成した画面をビデオ編集に利用することもできます。主な特長は次のとおりです。

- ①入力には、付属のマウスを使用します。
- ②メニューはわかりやすい絵文字（アイコン）表示。
- ③高解像度モード（タテ400ライン）でも、解像度モード（タテ200ライン）でも作画できます。ただし、作画をビデオ編集に利用する場合は、640×200ドット／320×200ドットのモードで作成してください。
- ④漢字を含む文字も簡単に表示できます。
- ⑤作画したデータをディスクに保存できます。
- ⑥秒単位に時間を指定して、作画を順次読み出して、ビデオ編集などに利用できます。
- ⑦作成した絵や文字をカラーインクジェットプリンタ10-700でコピーするためのユーティリティが付属しています。



# 付 録

# 付録

## A 1

## テキスト画面(V-RAM)へのアクセス

テキスト画面とその属性エリアは共に I/O ポートにあり、

テキスト画面が &H3000 ~ &H37FF (番地)

その属性が &H2000 ~ &H27FF (番地)

漢字属性が &H3800 ~ &H3FFF (番地)

の範囲で、それぞれ 2KB (2048 バイト) の容量をもっています。

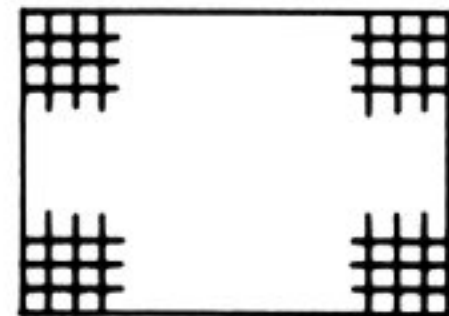
テキスト画面とその属性エリアとはちょうど 1 対 1 に順序よく対応していて、次に示すステートメントや関数を使って 1 バイト単位でアクセスすることができます。

出力用: POKE@、OUT

入力用: PEEK@、INP

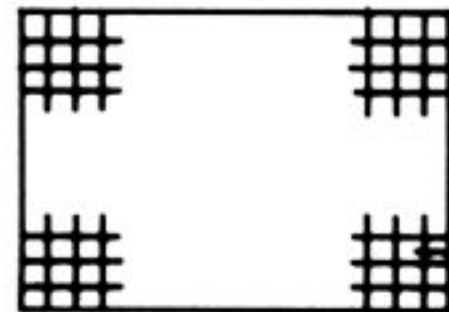
テキスト画面の属性 1 バイトのビット構成は下図のとおりです。

&H3000



テキスト画面

&H2000



テキスト属性

HS	VS	CG	CF	CR	G	R	B
----	----	----	----	----	---	---	---

以下、下位ビットから順次説明していきます。

### (1) B、R、G……色の指定をする (⇨COLOR)

テキスト画面の文字の色は B (青)、R (赤)、G (緑) の 3 ビットにより次のように決定されます。

ビット    2    1    0

G	R	B
---	---	---

0	0	0	……黒
0	0	1	……青
0	1	0	……赤
0	1	1	……マゼンタ (紫)
1	0	0	……緑



1 0 1 .....シアン（水色）

1 1 0 .....黄

1 1 1 .....白

(2) CR .....文字の反転モードの指定 (⇔CREV)

テキスト画面の文字の反転の指定はビット3で行ないます。

ビット3が0のとき標準モード、1のとき反転モードとなります。

(3) CF .....文字の明滅モードの指定 (⇔CF LASH)

テキスト画面の文字の明滅の指定は4ビットで行ないます。

ビット4が0のとき標準モード、1のとき明滅モードとなります。

(4) CG .....ROMCG / RAMCGの切り換えの指定 (⇔CGEN)

キャラクタ・ゼネレータには、ROMCGとRAMCGの2種類があり、それぞれ256文字まで表示させることができます。

テキスト画面にROMCGで生成した文字を表示するか、RAMCGで生成した文字を表示するかの指定は、ビット5で行ないます。

ビット5が0のときROMCG、1のときRAMCGを指定します。

(5) VS、HS …文字の拡大モードを設定します。(⇔CSIZE)

テキスト画面の文字のサイズの指定はビット6と7で行ないます。

ただし、ビット6（VS）に1を立てるときは、その横1行の属性ポートのビット6を立てなければなりません。

ビット 7 6

HS	VS
0	0
0	1
1	0
1	1

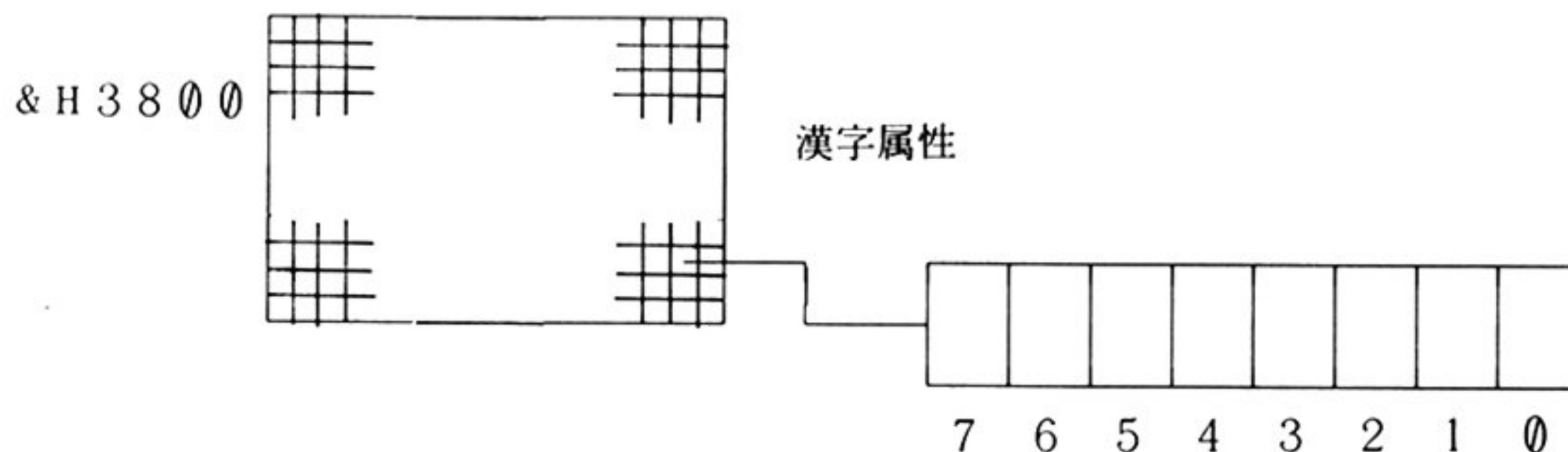
0 0 .....標準文字

0 1 .....たて2倍文字

1 0 .....横2倍文字

1 1 .....たて横2倍文字

漢字属性1バイトのビット構成は次のようになっています。



以下、テキスト属性同様、下位ビットから順次説明していきます。

- (1) 下位4ビット（ビット3，2，1，0）…漢字データ部の上位4ビットコードを示す。  
 全角文字（漢字ROM）が表示されているとき、漢字ROMアドレスの上位4ビットを示します。漢字ROMが選択されていない場合は無視されます。
- (2) ビット4……漢字ROMの第1水準か第2水準かの指定または、RAMCGのキャラクターモードか外字かの指定を行ないます。
- (3) ビット5……アンダーラインの指定  
 アンダーラインの指定は、ビット5で行ないます。  
 ここが0のときアンダーライン消法、1のときアンダーラインを表示します。
- (4) ビット6……全角文字のサイド指定  
 全角文字の場合ここがゼロならば文字の左側の部分、1ならば文字の右側部分であることを示しています。
- (5) ビット7……ROMCGと漢字ROMの選択の指定  
 漢字属性のビット4とビット7、およびテキスト属性のビット4とビット7、およびテキスト属性のビット5（CG）の値によって、ROMCG、RAMCG、漢字ROMのうちどれかが選択されるかが、決まります。

テキスト属性 ビット5	漢字属性		
	ビット4	ビット7	
1	0	0	RAMCG（キャラクターモード）
	0	1	RAMCG（外字モード）
	1	0	
0	1	1	
	0	0	ROMCG
	0	1	第1水準漢字ROM
	1	1	第2水準漢字ROM

## A 2

## 組み込み関数以外の数学的関数

組み込み数値関数にない三角関数と双曲線関数およびそれらの逆関数を定義する公式を示します。  
 使用する場合には各関数の定義域に注意する必要があります。



関数名	BASICの形式による公式	関数
secant	$A(X) = 1 / \cos(X)$	sec x
consecant	$B(X) = 1 / \sin(X)$	cosec x
cotangent	$C(X) = 1 / \tan(X)$	cot x
arcsine	$D(X) = \text{ATN}(X / \text{SQR}(1 - X^2))$	$\sin^{-1} x$
arccosine	$E(X) = -\text{ATN}(X / \text{SQR}(1 - X^2)) + \pi / 2$	$\cos^{-1} x$
arcsecant	$F(X) = \text{ATN}(\text{SQR}(X^2 - 1)) +$ $(\text{SGN}(X) - 1) * \pi / 2$	$\sec^{-1} x$
arccosecant	$G(X) = \text{ATN}(1 / \text{SQR}(X^2 - 1)) +$ $(\text{SGN}(X) - 1) * \pi / 2$	$\text{cosec}^{-1} x$
arccotangent	$H(X) = -\text{ATN}(X) + \pi / 2$	$\cot^{-1} x$
hyperbolic sine	$I(X) = (\text{EXP}(X) - \text{EXP}(-X)) / 2$	sinh x
hyperbolic cosine	$J(X) = (\text{EXP}(X) + \text{EXP}(-X)) / 2$	cosh x
hyperbolic tangent	$K(X) = -\text{EXP}(-X) / (\text{EXP}(X) +$ $\text{EXP}(-X)) * 2 + 1$	tanh x
hyperbolic secant	$L(X) = 2 / (\text{EXP}(X) + \text{EXP}(-X))$	sech x
hyperbolic cosecant	$M(X) = 2 / (\text{EXP}(X) - \text{EXP}(-X))$	cosech x
hyperbolic cotangent	$N(X) = \text{EXP}(-X) / (\text{EXP}(X) - \text{EXP}$ $(-X)) * 2 + 1$	cotanh x
arc-hyperbolic sine	$O(X) = \text{LOG}(X + \text{SQR}(X^2 + 1))$	$\sinh^{-1} x$
arc-hyperbolic cosine	$P(X) = \text{LOG}(X + \text{SQR}(X^2 - 1))$	$\cosh^{-1} x$
arc-hyperbolic tangent	$Q(X) = \text{LOG}((1 + X) / (1 - X)) / 2$	$\tanh^{-1} x$
arc-hyperbolic secant	$R(X) = \text{LOG}((\text{SQR}(1 - X^2) + 1) / X)$	$\text{sech}^{-1} x$
arc-hyperbolic cosecant	$S(X) = \text{LOG}((\text{SGN}(X) * \text{SQR}(X^2 + 1) + 1)$ $/ X)$	$\text{cosech}^{-1} x$
arc-hyperbolic cotangent	$T(X) = \text{LOG}((X + 1) / (X - 1)) / 2$	$\cot^{-1} x$

以上の各関数を使うときは、「DEF FN」ステートメントであらかじめプログラム中に定義しておく便利です。  
これらの関数は、パラメータXの定義域に十分注意して使ってください。

```
(例) 100 DEF FNA(X) = 1 / COS(X)
      . . . . .
      190 Y = . . .
      200 H = FNA(Y)
      . . . . .
```

## A 3

## 数値精度の変換

数値は、整数の場合、 $-32768 \sim 32767$ 、実数の場合、約  $-1.7 \times 10^{38} \sim 1.7 \times 10^{38}$  で、単精度型のとき有効数字16けたの精度で表現されます。

数値は、変数の属性文字(%, !, #)、型変数換の関数(CINT、CSNG、CDBL)によって、その精度を変換することができます。

精度の変換は、次の規則に従って行なわれます。

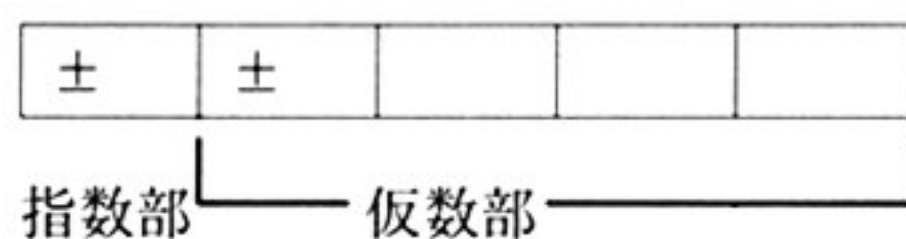
- (a) 数値が、別の精度の数値変数に代入されると、それは代入先の変数名の表わす精度で記憶されます。
- (例)  $A \% = 3.14 \dots\dots\dots A\%$  は整数型変数なので、 $A\%$  の値は 3 となります。
- (b) 数値をそれより低い精度の変数に代入すると、数値は丸められて低い精度で記憶されます。
- (例)  $A ! = 3.14159265358\# \dots A$  は単精度型変数なので、 $A$  の値は 3.1415927 に丸められます。
- (c) 数値をそれより高い精度の変数に代入すると、数値は高い精度の表示になりますが、より正確になることはありません。
- (例)  $A \# = 3.14 \dots\dots\dots A\#$  は倍数精度型変数なので、 $A\#$  の値は 3.1399999999664724 と表示されます。
- (d) 式の値を求めるときは、最も精度の高いオペランドと同じ精度に変換されます。
- (例)  $A \# = 4 \# / 7 \dots\dots\dots 4\#$  が倍精度型数値なので、計算の結果は倍精度型となり、 $A\#$  の値は、.5714285714285714 となります。
- (e) 関数の値は、単精度 8 けたが保証されますが、引数に倍精度型の数値が含まれると倍精度 16 けたまで求めることができます。
- (例)  $A = SQR(3) \dots\dots\dots A$  の値は、1.7320508  
 $A \# = SQR(3\#) \dots\dots\dots A\#$  の値は、1.732050807568877
- (f) 数式の値の型変数を行なうには、 $CINT$ 、 $CSNG$ 、 $CDBL$  の各関数を使います。  
 $CINT$  は値を整数型に、 $CSNG$  は単精度型に、 $CDBL$  は倍精度型にそれぞれ変換します。
- (例)  $A = CINT(R / 100) \dots\dots\dots R$  の値が 314 のとき、 $A$  の値は 3 になります。

## A 4

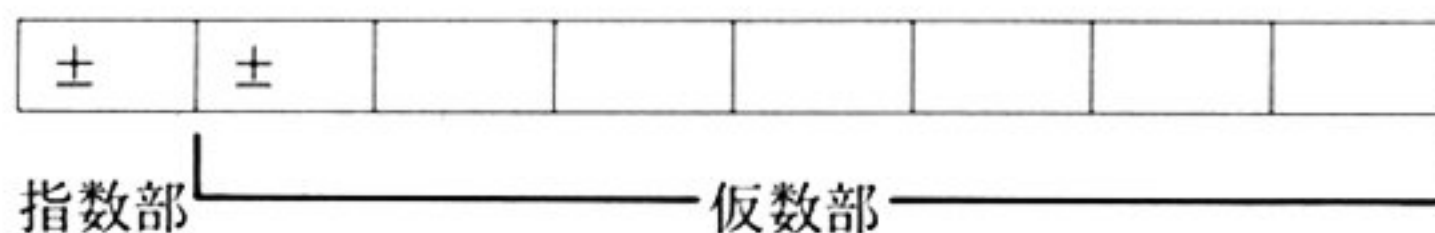
## 数値データの誤差

BASIC で扱う単精度型および倍精度型の数値は、コンピュータ内部では 2 進浮動小数点形式で表現され、演算や比較もこの形式のまま行なわれます。

単精度型 (5 バイト)



倍精度型 (8 バイト)



この内部表現上の制約のため、数値は必ずしも正確な値として記憶されとは限らず、このために起こる、演算結果および関係式の比較における誤差、画面やライン・プリンタなどの出力装置に表示される値とのずれ、などに注意する必要があります。

たとえば、単精度型や倍精度型の数値を使った演算の結果が、数学では整数となる場合でも、必ずしも整数が得られるとは限りません。

数値データの内部表現の誤差が表示される値に及ぼす影響について、簡単な例をあげて説明してみよう。



(例1)

```
10 X=0
20 FOR I=0 TO 1500
30 PRINT X;
40 X=X+.1
50 NEXT I
```

注) .1=0.1

(例1)は、変数Xの初期値を0として0.1ずつ加えては、その値を表示するというプログラムです。

このプログラムを実行すると、はじめは、0、.1、.2 ……と増えていきますが、46.1を過ぎると、46.1、46.199999、46.299999、…というように誤差が現れます。

これは、内部表現の誤差がたまったために生ずる現象です。

この不合理を解消する方法として、次の3つが考えられます。

(1) 整数の1を加え、それを10で割る。

```
10 X=0
20 FOR I=0 TO 1500
30 PRINT X/10;
40 X=X+1
50 NEXT I
```

(2) 10倍した値をCINTを使って整数型に変換してから10で割る。

```
10 X=0
20 FOR I=0 TO 1500
30 PRINT X;
40 X=X+.1
50 X=CINT(X*10)/10
60 NEXT I
```

(3) STR\$を使って文字型に変換してからVALを使って数値型に戻す。

```
10 X=0
20 FOR I=0 TO 1500
30 PRINT X;
40 X=X+.1
50 X=VAL(STR$(X))
60 NEXT I
```

(例2)

```
10 FOR I=0 TO 1 STEP .1
20 PRINT I;
30 NEXT I
```

(例2)は、FOR、NEXTループにおいてループ変数の初期値を0、増分を0.1として、終了値1まで回すというプログラムです。

このプログラムを実行すると、0、0.1、0.2 ……、0.9となって、最後の1までループしません。

これも、内部表現の誤差によって生ずる現象です。

これを防ぐためには、次のように、ループ変数の増分を整数にします。

```
10 FOR I=0 TO 10
```

```

20 PRINT I/10:
30 NEXT I

```

(例3)

```

10 X#=1/3*3
20 Y#=1
30 PRINT USING "###.#####";X#;Y#
40 IF X#=Y# THEN PRINT "X=Y#"

```

(例3)は、0.1を3で割って3倍した値をもつX#と、0.1の値をもつY#を比較して、等しければ「X#=Y#」と表示するプログラムです。

このプログラムを実行すると、「X#=Y#」と表示しません。これもまた、X#とY#との内部表現の誤差によって生じる現象です。

これを解消するためには、次のようにIF文の論理式の部分を変えて、10の精度で比較するようにします。

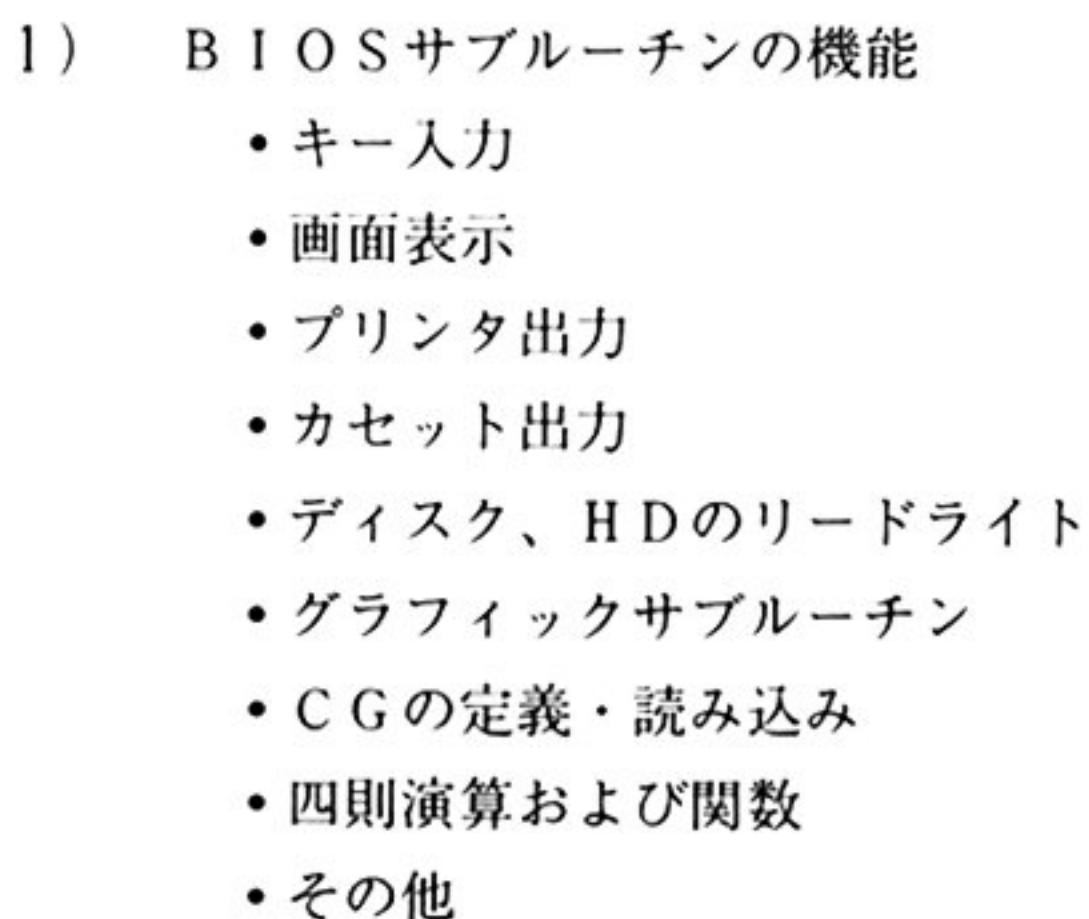
```

10 X#=1/3*3
20 Y#=1
30 PRINT USING "###.#####";X#;Y#
40 IF ABS(X#-Y#)<.00000001 THEN PRINT
   "X#=Y#"

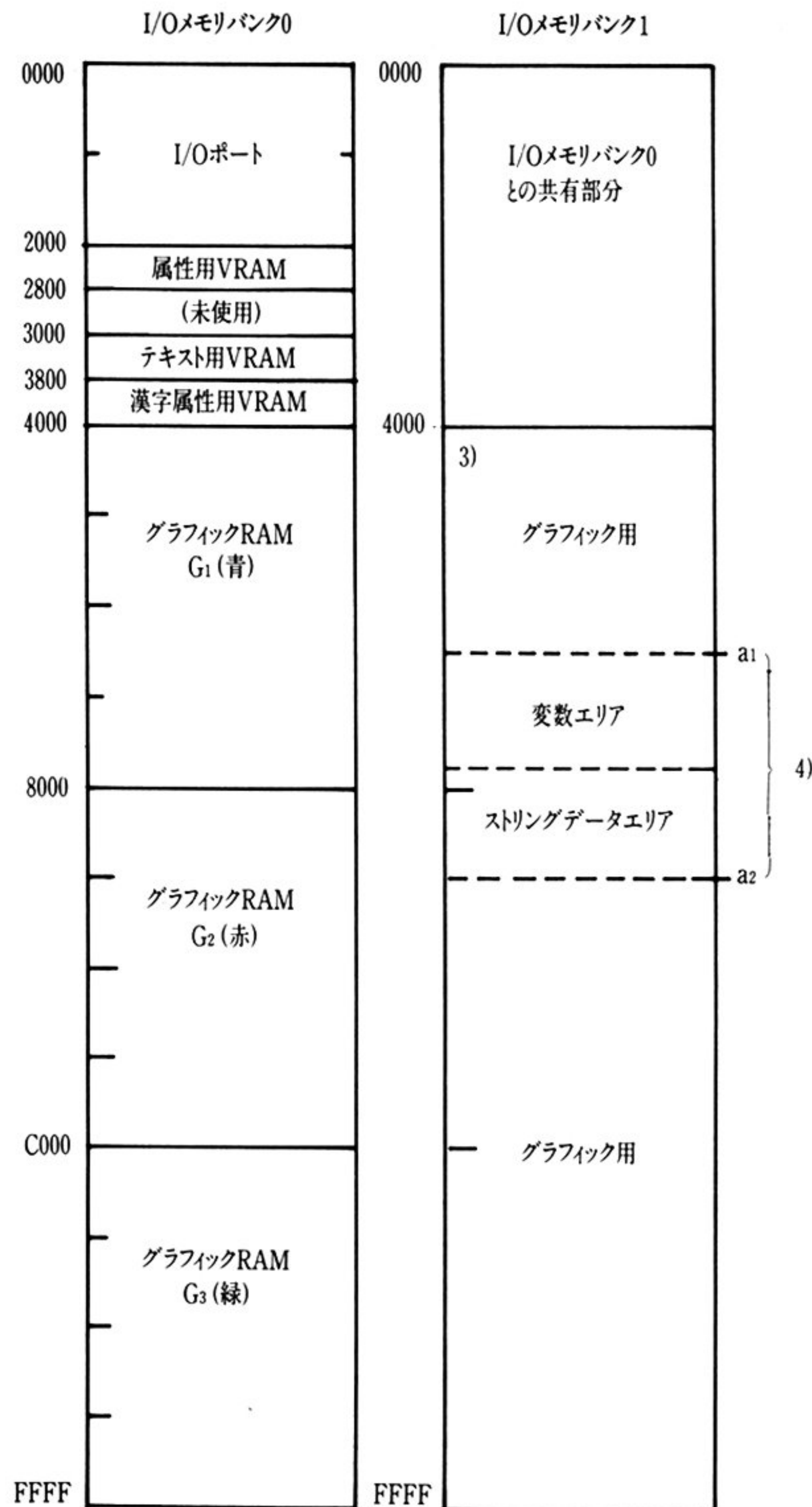
```



## メモリマップ<sup>o</sup>



— 230 —



3) OPTION SCREEN 1 および 2 の場合、VDIM 変数エリアとして使用することができる。

OPTION SCREEN 0 が設定されている場合は、バンク 0 と同様グラフィック表示用エリアとして使用できる。

4) VDIM 変数エリアとして使用する場合、VDIM CLEAR でその範囲を指定し、その外をグラフィック用として使用することができる。

a<sub>1</sub> は変数エリアの先頭アドレス、a<sub>2</sub> はエンドアドレス。VDIM CLEAR を実行していなければ 4000 ~ FFFF が変数エリアとして使用できる。



英数字、カナ、セミグラフィック文字、特殊文字などの図形文字は、小さな点が集まって構成されています。1つの文字を構成する点は、半角文字で $8 \times 8$ または $16 \times 8$ ドット、全角文字で $16 \times 16$ ドットあって、このドットパターンがROMCG (Read-Only Memory Character Generator) および漢字ROMに記憶されています。

これに対してRAMCG (Random Access Memory Character Generator) があり、ユーザーが好きなドットパターンをデザインして、これに記憶させておくことができます。

ユーザーがデザインできる図形文字（以下、ユーザー定義文字といいます）には、そのサイズによって、

- 〔1〕 たて $8 \times$ 横 $8$ ドット
- 〔2〕 たて $16 \times$ 横 $8$ ドット
- 〔3〕 たて $16 \times$ 横 $16$ ドット

の3つのタイプがあります。

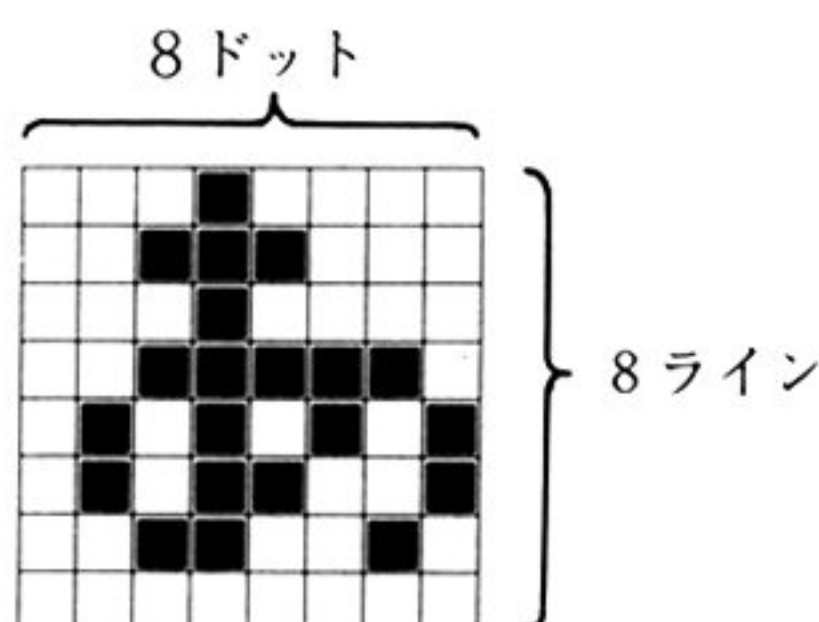
ここでは、その各々について作りかたと表示のしかたについて説明します。

#### 〔1〕 たて $8 \times$ 横 $8$ ドットの場合

たて $8 \times$ 横 $8$ ドットのユーザー定義文字を定義するには、24バイトの文字列が必要です。なぜならば、その文字列の表わすビットパターンが1個のユーザー定義文字を形成するからです。すなわち、1バイトのキャラクタコードのビットパターン8けたが、横 $8 \times$ たて1ドットパターンを表わし、8バイトの文字列でたて8ラインのドットパターンを構成しています。これが3原色の合成原理により、青、赤、緑の3画面分必要なので、結局、8バイト $\times$ 3画面=24バイトの文字列が必要となります。

この24バイトの文字列のうち、最初の8バイトが青、次の8バイトが赤、最後の8バイトが緑の部分のドットパターンを表わしています。

たとえば、ひらがなの「あ」という文字を青地にシアン色で作る場合についてみてみましょう。



■はドットがセットされていることを示す。

ユーザー定義文字は、 $8 \times 8$ ドットパターンで表わされ、ひらがなの「あ」は上のようなパターンになります。これを2進数のパターンで表わし、さらに16進数表現に変換すると、次のようになります。

(2進数表現)		(16進数表現)
0 0 0 1 0 0 0 0	⇒	1 0 .....①
0 0 1 1 1 0 0 0	⇒	3 8 .....②
0 0 0 1 0 0 0 0	⇒	1 0 .....③
0 0 1 1 1 1 1 0	⇒	3 E .....④

0 1 0 1 0 1 0 1	⇒	5 5	.....⑤
0 1 0 1 1 0 0 1	⇒	5 9	.....⑥
0 0 1 1 0 0 1 0	⇒	3 2	.....⑦
0 0 0 0 0 0 0 0	⇒	0 0	.....⑧

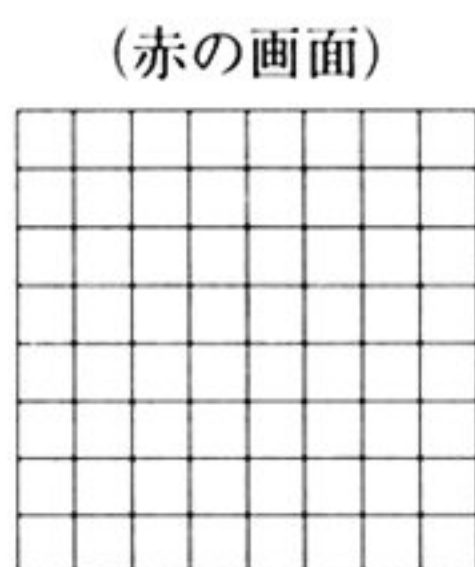
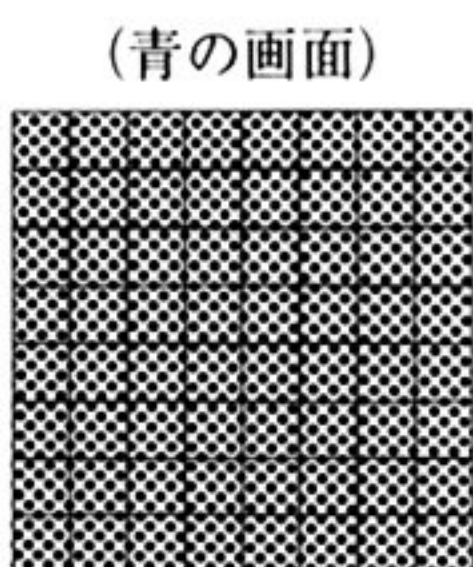
したがって、このドットパターンは、

HEXCHR\$ ("10 38 10 3E 55 59 32 00")

① ② ③ ④ ⑤ ⑥ ⑦ ⑧

となります。

シアン色は青と緑を合成して得られ、青地にシアン色の「あ」という文字を作るためには、青の画面、赤の画面、緑の画面に対するパターンをそれぞれ次のように定義しなければなりません。



(青の画面)	(赤の画面)	(緑の画面)
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 0 1 0 0 0 0 ⇒ 1 0
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 1 1 1 0 0 0 ⇒ 3 8
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 0 1 0 0 0 0 ⇒ 1 0
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 1 1 1 1 1 0 ⇒ 3 E
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 1 0 1 0 1 0 1 ⇒ 5 5
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 1 0 1 1 0 0 1 ⇒ 5 9
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 1 1 0 0 1 0 ⇒ 3 2
1 1 1 1 1 1 1 1 ⇒ F F	0 0 0 0 0 0 0 0 ⇒ 0 0	0 0 0 0 0 0 0 0 ⇒ 0 0

したがって、青地にシアン色の「あ」を定義するのに必要な文字列は、

HEXCHR\$ ("FFFFFFFFFFFFFFFF00000000000000001038103E55593200")

となります。

このサイズのパターンを定義できるコード（キャラクタコード）は0～255で、256個の文字を定義することができます。たとえばこれをコード32に定義するにはDEFCHR\$ステートメントを使って、

DEFCHR\$ (32) = HEXCHR\$ ("FFFFFFFFFFFFFFFF00000000000000001038103E55593200")と書きます。

こうして定義した文字は、CGEN1とPRINT#0ステートメントを使って画面に表示することができます。

```
(例) 100 KMODE 0
      110 CGEN1
      120 A$ = "FFFFFFFFFFFFFFFF00000000000000001038103E55593200"
      130 DEFCHR$ (32) = HEXCHR$ (A$)
      140 PRINT#0, CHR$ (32)
      150 CGEN0
```



青の画面、赤の画面、緑の画面の3枚とも同じデータの場合は、8バイトの文字列ですませることができます。

たとえば、黒地に白の「あ」を定義するには、

```
DEFCHR$ (32) = HEXCHR$ ("1038103E555932001038103E555932001038103E55593200")
```

としないで、

```
DEFCHR$ (32) = HEXCHR$ ("1038103E55593200")
```

とするだけですみます。

〔2〕 たて16×横8ドットのユーザー定義文字の定義には、8×8ドットの文字がたて2倍にのびたパターンなので、48バイトの文字列が必要です。

(青の画面)

1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							

(赤の画面)

17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							

(緑の画面)

33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							

16×8ドットのユーザー定義文字は、16×8ドットパターンで表わされます。ここにデザインしたパターンを8×8のときと同様に2進数のパターンで表わし、さらに16進数表現に変換してコードの文字列にするのです。

このサイズのパターンを定義できるコードは&H100、&H102、&H104、……、&H1FEの128文字分です。たとえばあるパターンをコードの&H100に定義するには、

```
DEFCHR$ (&H100) = HEXCHR$ ("HHH………HHH")
```

96けた(48バイト)

と書きます。

こうして定義した文字は、CGENとPRINT#0ステートメントを使って画面に表示することができます。

```
110 CGEN 2
```

```
120 A$ = "HHH………HHH" → 48バイトの任意の文字列
```

```
130 DEFCHR$ (&H100) = HEXCHR$ (A$)
```

```
140 PRINT#0, CHR$ (0)
```

※    &H100のコードは    PRINT#0, CHR\$(0)  
       &H102のコードは    PRINT#0, CHR\$(&H2)  
       &H104は                PRINT#0, CHR\$(&H4)  
       .....

というように書いて表示します。

なお、青の画面、赤の画面、緑の画面の3枚とも同じデータの場合は、16バイトの文字列ですませることができます。

〔3〕 たて16×横16ドットの場合

たて16×横16ドットのユーザー定義文字の定義には、96バイトの文字列が必要です。

(青の画面)

1																17
2																18
3																19
4																20
5																21
6																22
7																23
8																24
9																25
10																26
11																27
12																28
13																29
14																30
15																31
16																32



(赤の画面)

33																49
34																50
35																51
36																52
37																53
38																54
39																55
40																56
41																57
42																58
43																59
44																60
45																61
46																62
47																63
48																64

(緑の画面)

65																81
66																82
67																83
68																84
69																85
70																86
71																87
72																88
73																89
74																90
75																91
76																92
77																93
78																94
79																95
80																96

16×16ドットのユーザー定義文字は、16×16のドットパターンで表わされます。ここにデザインしたパターンを、8×8のときと同様に2進数のパターンで表わし、さらに16進数表現に変換してコードの文字列にします。

このサイズのパターンを定義できるコードは&J7621～&J7660の64文字分です。たとえば、あるパターンをコードの&J7621に定義するには、

```
DEFCHR$ (&J7621) = HEXCHR$ (" HHH...HHH ")
                                192けた (96バイト)
```

と書きます。

こうして定義した文字は、KMODE1を実行後、普通のPRINT文を使って画面に表示することができます。

```
KMODE1
PRINT CHR$ (&J7621)
```

なお、青の画面、赤の画面、緑の画面の3枚とも同じデータの場合は、32バイトの文字列ですませることができます。











# タチバナ株式会社

本社 〒545 大阪市阿倍野区長池町22番22号  
 電話 06 (621) 1221 (大代表)  
 電子機器事業本部 〒329-21 栃木県矢板市早川町174番地  
 電話 02874 (3) 1131 (大代表)

お客様へ……お買いあげ年月日、お買いあげ店名を記入されますと、修理などの依頼のときに便利です。

お買いあげ年月日	年      月      日
お買いあげ店名	
	電話番号
もよりの お客様ご相談窓口	
	電話番号